REDUCING THE COMMUNICATION COST OF FEDERATED LEARNING THROUGH MULTISTAGE OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A central question in federated learning (FL) is how to design optimization algorithms that minimize the communication cost of training a model over heterogeneous data distributed across many clients. A popular technique for reducing communication is the use of *local steps*, where clients take multiple optimization steps over local data before communicating with the server (e.g., FedAvg, SCAFFOLD). This contrasts with *centralized methods*, where clients take one optimization step per communication round (e.g., Minibatch SGD). A recent lower bound on the communication complexity of first-order methods shows that centralized methods are optimal over highly-heterogeneous data, whereas local methods are optimal over purely homogeneous data (Woodworth et al., 2020b). For intermediate heterogeneity levels, no algorithm is known to match the lower bound. In this paper, we propose a multistage optimization scheme that nearly matches the lower bound across all heterogeneity levels. The idea is to first run a local method up to a heterogeneity-induced error floor; next, we switch to a centralized method for the remaining steps. Our analysis explains empirically-successful stepsize decay methods in FL (Charles & Konečnỳ, 2020; Reddi et al., 2020). We demonstrate the scheme's practical utility in a variety of image classification tasks.

1 INTRODUCTION

In federated learning (FL) (McMahan et al., 2017; Kairouz et al., 2019; Li et al., 2020), distributed clients interact with a central server to learn a model without directly sharing their data with the server. The training objective is to solve the following minimization problem:

$$\min_x \Big[F(x) = \frac{1}{N}\sum_{i\in[N]} F_i(x)\Big] \tag{1}$$

where variable x is the model parameter, i indexes the clients (or devices), N is the number of clients, and $F_i(x)$ is a (strongly convex) loss that only depends on that client's data. Typical FL deployments have two properties that make optimizing Eq. (1) challenging: (i) *Data heterogeneity:* Let $F_i(x) = \mathbb{E}_{z_i\sim P_i}[f(x; z_i)]$ for some loss function f, where z_i is the client's data and P_i's capture the heterogeneity of the clients' data distributions. Formally, define client heterogeneity ζ as

$$\zeta^2 := \sup_x \frac{1}{N}\sum_{i\in[N]} \|\nabla F(x) - \nabla F_i(x)\|^2, \tag{2}$$

which captures how different the local client gradients are from the overall global gradient (in the worst-case) (Woodworth et al., 2020a). The heterogeneity at the optimum $x^* := \arg\min_x F(x)$ is also commonly considered, i.e. $\zeta_*^2 := (1/N)\sum_{i\in[N]} \|\nabla F_i(x^*)\|^2$. (ii) *Communication cost:* In many deployments of FL, clients have limited bandwidths (e.g., mobile devices) and hence high communication cost. Due to these two challenges, most federated optimization algorithms alternate between local rounds of computation, where clients process only their own data to save communication, and global rounds, where clients synchronize with the central server to resolve their disagreements due to heterogeneity.

Several federated algorithms navigate this trade-off between reducing communication and resolving data heterogeneity by modifying the amount and the nature of *local and global computation* (McMahan et al., 2017; Li et al., 2018; Wang et al., 2019b;a; Li et al., 2019; Karimireddy et al., 2020b;a; Al-Shedivat et al., 2020; Reddi et al., 2020; Charles & Konečnỳ, 2020; Mitra et al., 2021). Recently, Woodworth et al. (2020a) showed a lower bound on the convergence rate (with respect to number of communication rounds R) of any first-order federated optimization algorithms for strongly-convex, smooth objectives as a function of client data heterogeneity ζ_* (Section 2, Eq. (6)). This lower bound highlights two gaps in the literature: (1) Today, no single optimization algorithm has been shown to match this lower bound for *all* values of ζ_*. (2) Although different baseline optimization algorithms match the lower bound in different heterogeneity regimes, it remains unknown if *any* algorithm can match the lower bound when $0 < \zeta_*^2 < \beta^{3/2}\Delta/\mu^{1/2}$, where Δ is the initial function suboptimality (see Assumption 6), and the functions F_i are β-smooth and μ-strongly convex (see App. C for definitions of smoothness and strong convexity). Δ can be sizable if one starts optimization far away from the optimum, and the gap between the upper and the lower bound in this regime is significant (§ 2). To explain these gaps more precisely, we first examine several baseline optimization algorithms.

To begin, consider the simple baseline of Minibatch Stochastic Gradient Descent (SGD) (Woodworth et al., 2020a, shortened as SGD). Here, in each global communication round, the central server sends each device the current model. Then each device calculates a single local stochastic gradient at this model over a minibatch of size K and sends it to the server. The server averages these to make a global minibatch gradient over NK samples and performs a model update (Algorithm 2). Further, this approach can be accelerated in the Nesterov sense (Ghadimi & Lan, 2012, AC-SA). Woodworth et al. (2020a) recently showed that Accelerated Minibatch SGD (shortened as AC-SA) is nearly optimal (up to exponentially shrinking condition numbers factors in communication rounds R) in the highly-heterogeneous regimes when $\zeta_*^2 > \beta^{3/2}\Delta/\mu^{1/2}$. However, in other regimes, AC-SA may be suboptimal as it does not exploit local computation to reduce communication.

A popular alternative exploiting local computation is Federated Averaging (FedAvg) (McMahan et al., 2017)[1] (Algorithm 6). Here each client i independently runs SGD on its own objective F_i for K iterations (local updates), and periodically communicates their respective most recent iterate to the server, which then averages these iterates. Compared to (Accelerated) SGD, this approach allows clients to learn from local data in between communication rounds. However, on strongly convex objectives, FedAvg has only been shown to achieve the lower bound when the data is homogeneous, i.e. $\zeta_*^2 = \zeta^2 = 0$. In fact, FedAvg's known rates can only outperform AC-SA's rates when heterogeneity is almost insignificant: $\zeta^2 \leq \mu\epsilon$, where ϵ is the target sub-optimality gap for Eq. (1) (Woodworth et al., 2020a) (for more details on ϵ see Definition 1). Further, an algorithm specific lower bound shows that FedAvg is provably sub-optimal; improving the analysis will not bring FedAvg meaningfully closer to optimal rates or even allow FedAvg to beat AC-SA when heterogeneity is non-negligible Woodworth et al. (2020a). Several approaches have been proposed to improve FedAvg via cross-device variance reduction (Karimireddy et al., 2020b; Gorbunov et al., 2020) and adaptivity and momentum (Reddi et al., 2020). Empirically, these variants of FedAvg can outperform baselines like (Accelerated) SGD. However, such gain is not explained by existing analyses.

In summary, currently there are only two baseline optimization algorithms that (nearly, up to exponentially shrinking condition numbers) theoretically match the lower bound of Woodworth et al. (2020a) in some heterogeneity regime: FedAvg when $\zeta^2 = 0$ (i.e., $\zeta_*^2 = 0$) and AC-SA when $\zeta_*^2 > \beta^{3/2}\Delta/\mu^{1/2}$. Several natural questions follow. (1) Does there exist an algorithm that achieves the lower bound when $0 < \zeta_*^2 < \beta^{3/2}\Delta/\mu^{1/2}$? In fact, Woodworth et al. (2020a) posed a weaker open question, asking whether it was possible to improve on AC-SA in the regime where data heterogeneity was bounded but not insignificant. (2) If so, can the *same* algorithm simultaneously achieve optimal convergence rates across *all* values of ζ_*?

This Work. We answer both questions affirmatively when $\zeta \simeq \zeta_*$ by drawing on *multistage algorithms*, in which learning rates decay during optimization. Such algorithms have been used to achieve fast convergence rates in convex optimization (Aybat et al., 2019; Fallah et al., 2020). In the federated setting, they have heuristically improved practical performance (Reddi et al., 2020; Charles & Konečnỳ, 2020).

[1]This approach is also referred to as Local Stochastic Gradient Descent (Stich, 2018; Khaled et al., 2020; Woodworth et al., 2020b), LocalUpdate (Charles & Konečnỳ, 2020).

Our work exploits the following key insight. Far from the optimum, local computation can be beneficial for reducing communication; near the optimum, it is more important to communicate frequently (per model update) to resolve heterogeneity. This intuition has been explored in the homogeneous setting (Wang & Joshi, 2019), where the theoretical benefits do not appear to give order-wise convergence improvements.

Contributions. In this paper, we first show that multistage optimization, when applied to baseline federated algorithms, achieves comparable or better worst-case convergence rates than known baselines (including SGD and FedAvg) in all heterogeneity regimes. In doing so, multistage algorithms resolve an open problem from (Woodworth et al., 2020a), which asked if one can design an algorithm that combines the advantages of both FedAvg and SGD and enjoys guarantees that dominate both. We propose multistage algorithms that (nearly) match the lower bound of Woodworth et al. (2020a) on convergence rates for first-order federated algorithms.

We show these theoretical gains by first analyzing an algorithm that (nearly) achieves the lower bound: a simple two-stage combination of ASG (a variant of AC-SA in Theorem 6) and FedAvg. Loosely, we first run FedAvg up to an "optimization error floor" that depends on client heterogeneity; then, we use ASG to complete the optimization. Further, we show that similar gains can be obtained by a more general multistage algorithm which gradually moves from the FedAvg update to ASG update. This multistage has the added benefit of not requiring the knowledge of the noise bound σ^2, the heterogeneity ζ^2, or the initial distance to the optimum Δ. This intuition is not specific to the combination of ASG and FedAvg. We demonstrate the generality of multistaging by applying it to other pairs of FL optimization algorithms to obtain the same nearly-optimal worst-case convergence (but possibly much better average-case rates, § 4). Our theoretical results may help explain the empirical success of stepsize scheduling in FL (Charles & Konečnỳ, 2020; Reddi et al., 2020) (App. A). Finally, we demonstrate the practical value of these theoretical insights through both convex and nonconvex experiments in § 4.

2 Model and Related Work

We assume that each F_i is μ-strongly convex (defined in Assumption 1) and β-smooth (defined in Assumption 2). We let $\kappa = \beta/\mu$ denote the condition number. N is the number of clients. Heterogeneity ζ is defined in Assumption 3 and heterogeneity at optimum ζ_* is defined in Assumption 4. We assume the client gradient variance is upper bounded as σ^2, formally defined in Assumption 5. We bound the initial suboptimality gap as Δ (see Assumption 6).

Optimization proceeds in rounds; K is the number of client iterations per communication round, and R is the number of communication rounds. In some cases, clients take minibatches of size B in their local steps; unless specified otherwise, B is 1. Our goal is to minimize the sub-optimality error $\mathbb{E}F(\hat{x}) - F(x^*)$ (where $\hat{x}$ is the solution returned by an optimization algorithm) after R rounds (or to minimize *communication round complexity*, the number of rounds R to a given error ϵ). R is a standard proxy for the communication cost. We use the notation $\tilde{\mathcal{O}}$ to hide polylogarithmic factors.

2.1 Related Work

To date, the fastest (worst-case) convergence rate in terms of R for our setting is achieved either by FedAvg (McMahan et al., 2017) or SGD in different heterogeneity regimes.

FedAvg. The convergence of FedAvg for convex minimization was first studied in the homogeneous client setting (Stich, 2018; Wang & Joshi, 2018; Woodworth et al., 2020b). These rates were later extended to the heterogeneous client setting (Khaled et al., 2020; Karimireddy et al., 2020b; Woodworth et al., 2020a; Koloskova et al., 2020), including a lower bound (Woodworth et al., 2020a). The fastest known rate for the worst-case sub-optimality gap of FedAvg is due to Gorbunov et al. (2020) (Table 1):

$$\tilde{\mathcal{O}}(\kappa\Delta\exp(-\frac{KR}{\kappa}) + \frac{\sigma^2}{\mu NKR} + \frac{\kappa\sigma^2}{\mu KR^2} + \frac{\kappa\zeta^2}{\mu R^2}). \tag{3}$$

If we take enough local steps K, this error is dominated by the last term, $\frac{\beta\zeta^2}{\mu^2 R^2}$. If heterogeneity is low (e.g., $\zeta^2 = 0$), then for large enough K, only one round of communication is needed. This explains why FedAvg outperforms AC-SA in the low-heterogeneity regime.

Table 1: Convergence rates of sub-optimality gap under strongly convexity, hiding constants and log terms. ASG is a variant of the Accelerated Minibatch SGD from Theorem 6. When applicable, K is taken large enough to match the dominating terms in R . "X $\rightarrow$ Y" denotes first running "X" and using the output as input to "Y". All rates achieved by FedAvg $\rightarrow$X for some algorithm X are also achieved by SCAFFOLD$\rightarrow$X. The two SCAFFOLD rates are from two different stepsizes.

Method/Analysis	**Sub-optimality rate:** $\mathbb{E}F(\hat{x}) - F(z^*) \leq \tilde{\mathcal{O}}(\cdot)$
Minibatch Methods	
SGD(Woodworth et al., 2020a)	$\frac{\beta\Delta}{\mu}\exp(-\frac{R}{\kappa}) + \frac{\sigma^2}{\mu NKR}$
AC-SA (Woodworth et al., 2020a)	$\Delta\exp(-\frac{R}{\sqrt{\kappa}}) + \frac{\sigma^2}{\mu NKR} \quad [R \geq \Omega(\sqrt{\kappa})]$
Federated Algorithms	
SCAFFOLD (Karimireddy et al., 2020b)	$(\beta\Delta + \frac{\mu\zeta_*^2}{\beta^2})\exp(-\frac{R}{\kappa}) + \frac{\sigma^2}{\mu NKR}$
FedAvg (Gorbunov et al., 2020)	$\kappa\Delta\exp(-\frac{KR}{\kappa}) + \frac{\sigma^2}{\mu NKR} + \frac{\beta\sigma^2}{\mu^2 KR^2} + \frac{\beta\zeta^2}{\mu^2 R^2}$
SCAFFOLD (Theorem 10)	$\kappa\Delta\exp(-\frac{KR}{\kappa}) + \frac{\sigma^2}{\mu NKR} + \frac{\beta\sigma^2}{\mu^2 KR^2} + \frac{\beta\zeta^2}{\mu^2 R^2}$
FedAvg $\rightarrow$ SGD (Theorem 4)	$\min\{\kappa\Delta, \frac{\beta^2\zeta^2}{\mu^3 R^2}\}\exp(-\frac{R}{\kappa}) + \frac{\sigma^2}{\mu NKR}$
FedAvg $\rightarrow$ ASG (Theorem 5)	$\min\{\Delta, \frac{\beta\zeta^2}{\mu^2 R^2}\}\exp(-\frac{R}{\sqrt{\kappa}}) + \frac{\sigma^2}{\mu NKR} + \frac{\beta\sigma^2}{\mu^2 NKR^3}$
Lower bound (Woodworth et al., 2020a)	$\min\{\frac{\Delta}{\sqrt{\kappa}}, \frac{\mu\zeta_*^2}{\beta^2}\}\exp(-\frac{R}{\sqrt{\kappa}}) + \frac{\sigma^2}{\mu NKR}$

Accelerated SGD (AC-SA). With highly heterogeneous clients, AC-SA is (nearly, up to exponentially shrinking condition numbers) optimal, with a convergence rate of (Ghadimi & Lan, 2012; Woodworth et al., 2020a)

$$\mathcal{O}(\Delta\exp(-\frac{R}{\sqrt{\kappa}}) + \frac{\sigma^2}{\mu NKR}) \ . \tag{4}$$

Recall the FedAvg upper bound in Eq 3. Consider the noiseless case: $\sigma^2 = 0$. By solving for the round complexity needed for FedAvg's upper bound and AC-SA's upper bound to reach ϵ accuracy, we see FedAvg's communication R complexity is not less than that of the basic minibatch algorithms unless $\zeta^2 \leq \mu\epsilon\log(\Delta/\varepsilon)$, i.e., heterogeneity must scale with ϵ. This is not an artifact of loose analysis as Woodworth et al. (2020a) provide the following lower bound for the convergence rate of FedAvg:

$$\Omega(\min\{\Delta\exp(\frac{-R}{\kappa}), \frac{\kappa\zeta_*^2}{\mu R^2}\} + \frac{\sigma^2}{\mu NKR} + \min\{\Delta, \frac{\kappa\sigma^2}{\mu K^2 R^2}\}). \tag{5}$$

Considering the noiseless case, AC-SA has a convergence rate of $\Delta\exp(-R/\sqrt{\kappa})$ and FedAvg has the lower bound $\min\{\Delta\exp(-R/\kappa), \kappa\zeta_*^2/\mu R^2\}$. $\Delta\exp(-R/\kappa)$ is always worse than $\Delta\exp(-R/\sqrt{\kappa})$, so FedAvg can only ever outperform AC-SA when $\Delta\exp(-R/\sqrt{\kappa})$ requires a larger R than $\kappa\zeta_*^2/\mu R^2$ to reach ϵ accuracy. Solving for R, this means that FedAvg's lower bound can only result in lower communication complexity than AC-SA when $\zeta_*^2 \leq \mu\epsilon$. So even in the absolute best case scenario, where the lower bound of (Woodworth et al., 2020a) is the convergence rate of FedAvg, heterogeneity still must scale with ϵ for it to outperform minibatch algorithms.

Lower Bound. The lower bound for first-order distributed algorithms in the heterogeneous setting, from (Woodworth et al., 2020a), is (also shown in Table 1):

$$\Omega(\min\{\frac{\Delta}{\sqrt{\kappa}}, \frac{\mu\zeta_*^2}{\beta^2}\}\exp(-\frac{R}{\sqrt{\kappa}}) + \frac{\sigma^2}{\mu NKR}) \ . \tag{6}$$

Note the exponential difference between the lower bound in Eq. (6) and the tightest-known ζ-dependent upper bound (i.e., FedAvg, Eq. (3)) with respect to their dependency on heterogeneity ζ^2 and rounds R. In the upper bound (Eq. (3)), ζ^2 is scaled by $1/R^2$, whereas in the lower bound (Eq. (6)), it is scaled by a term decaying exponentially fast in R assuming $\zeta = \mathcal{O}(\zeta^*)$. A natural question is whether this gap can be closed.

Closing the gap. Currently, no algorithm is known to close this gap, or even achieve comparable or faster rates than *both* AC-SA and FedAvg across all heterogeneity regimes. This may be because prior theoretical analysis focused mainly on three hyperparameter regimes: (i) local update stepsizes

scale as $1/K$ (e.g., S-Local-SVRG (Gorbunov et al., 2020), FedAdam (Reddi et al., 2020), FedLin (Mitra et al., 2021)), (ii) local update stepsizes are small or zero (e.g., SCAFFOLD), and (iii) local update stepsizes do not depend on K (e.g., FedAvg). Intuitively, regimes (i) and (ii) do not take sufficient advantage of local steps to excel in the moderate- to low-heterogeneity regime, whereas algorithms in regime (iii) may converge too quickly to local optima, leading to a severe dependency on heterogeneity. The only work outside these regimes (to our knowledge) applies only to quadratic objective functions (Charles & Konečnỳ, 2020) and does not achieve the FedAvg rate, possibly due to insufficiently aggressive stepsize decay.

In this paper, we show that this theoretical gap can be closed by carefully combining existing methods (in stepsize regimes which have known analyses) in *stages*. This multi-stage approach sequentially leverages the large local stepsizes of (iii) and the small local stepsizes of (i) and (ii), thus getting the benefits of both (Table 1).[2] In the next section, we explain how to achieve these gains through a simple multistage federated minimization framework.

Multistage Algorithms. Multistage algorithms have been employed in the classical convex optimization setting to obtain optimal rates (Aybat et al., 2019; Fallah et al., 2020; Ghadimi & Lan, 2012; Woodworth et al., 2020a). The idea is to exponentially decrease the learning rate of an algorithm (e.g., AC-SA) in stages to balance the optimizer's error due to bias and variance. Our federated setting differs from the prior multistage optimization in the following ways: (1) there is optimization error due to client drift (Reddi et al., 2020; Karimireddy et al., 2020b) in addition to error due to bias and variance; this additional type of error cannot be accounted for by existing multistage algorithms, (2) our objective is to minimize communication rounds rather than iteration complexity, and (3) we "multistage" or schedule more hyperparameters than just a learning rate.

Wang & Joshi (2019) proposed a strategy with decreasing number (K) of local updates to transition from FedAvg to SGD, so as to speed up distributed optimization. This is conceptually similar to the learning rate decay of multistage algorithms studied in our work. However: (1) their work focuses on reducing wall-clock time in the *homogeneous* client setting whereas we focus on reducing communication cost among heterogeneous clients. (2) Further Wang & Joshi (2019) do not show order-wise improvements in convergence rate. In the homogeneous setting, a more recent work showed that a simple variant of AC-SA already achieves the optimal worst-case convergence rate without using any learning rate decay or multistage (Woodworth et al., 2021).

3 MULTISTAGE FEDERATED MINIMIZATION

In this section, we introduce a framework for multistage federated minimization algorithms in the strongly convex setting. The framework (specified in Algorithm 1) proceeds over S stages; between stages, we can change both the optimization algorithm and the associated hyperparameters (e.g., stepsizes). Each stage $s \leq S$ requires us to choose a local (client) optimization algorithm (e.g., FedAvg), a set of inner (client) and outer (server) stepsizes, a duration in number of rounds R, a number of local client steps K, and other hyperparameters (e.g., momentum, client weights).

Local updates: For a given stage $s \leq S$, we iterate over rounds $r \in \{1, \ldots, R\}$. In a round r, client i first initializes their iterate as $x_{i,0}^{(s,r)} = x^{(s,r)}$, and then runs K local update steps of its local optimizer using the local stepsize $\eta_l^{(s)}$. Next we give the different update rules used by different algorithms at client i. Here $\tilde{\nabla} F_i(x)$ denotes the possibly random gradient evaluated at x.

FedAvg (Algorithm 6) and (Accelerated) SGD (Algorithms 2 and 3) at step k use the update rule:

$$x_{i,k}^{(s,r)} \leftarrow x_{i,k-1}^{(s,r)} - \eta_l^{(s,r)} \tilde{\nabla} F_i(x_{i,k-1}^{(s,r)}) \tag{7}$$

However, (Accelerated) SGD sets the local stepsize as $\eta_l^{(s,r)} = 0$. This implies that $x_{i,k}^{(s,r)} = x^{(s,r)}$ for all $0 \leq k \leq K$.

[2]Note that in Table 1, $\min$ in the FedAvg $\rightarrow$SGD and FedAvg $\rightarrow$ASG rates comes from using a meta-algorithm that runs SGD and ASG respectively if the first part of the $\min\{\cdot\}$ is smaller. The $\min\{\cdot\}$ in the M-FedAvg $\rightarrow$M-ASG rate comes from running M-ASG and M-FedAvg $\rightarrow$M-ASG in parallel and then evaluating objective values of both solutions and choosing the best one (this requires one extra round of communication).

Algorithm 1 Multistage federated minimization framework

Input: Number of stages S, round lengths $\{R^{(s)}\}_{s=1}^{S}$, inner stepsizes $\{\{\eta_l^{(s,r)}\}_{r=1}^{R^{(s)}}\}_{s=1}^{S}$, outer stepsizes $\{\{\eta_g^{(s,r)}\}_{r=1}^{R^{(s)}}\}_{s=1}^{S}$, momentums $\{\gamma^{(s)}\}_{s=1}^{S}$, initial point $x^{(0)}$, weights for calculating weighted average solution from iterates $\{\{\{w_k^{(s,r)}\}_{k=1}^{K}\}_{r=1}^{R^{(s)}}\}_{s=1}^{S}$
for $s = 1, \ldots, S$ **do**
  $x^{(s,0)}, x^{(s,1)} \leftarrow x^{(s-1)}$
  for $r = 1, \ldots, R^{(s)}$ **do**
    ▷ **Broadcast the iterate from current round**
    $x^{(s,r)} \leftarrow x^{(s)}$
    ▷ **Obtain pseudo-gradient from clients**
    $G^{(s,r)} \leftarrow \mathsf{ClientOPT}(\eta_l^{(s,r)}, x^{(s,r)})$
    ▷ **Perform server update**
    $x^{(s,r+1/2)} \leftarrow (1+\gamma^{(s)})x^{(s,r)} - \gamma^{(s)}x^{(s,r-1)}$
    $x^{(s,r+1)} \leftarrow x^{(s,r+1/2)} - \eta_g^{(s,r)}G^{(s,r)}$
  end for
  $x^{(s)} \leftarrow \mathsf{StageWeightedAverage}(\{\{w_k^{(s,r)}\}_{k=1}^{K}\}_{r=1}^{R^{(s)}})$
end for
return $x^{(S)}$

Client variance-reduced methods like SS-Local-SGD (Gorbunov et al., 2020) (which is a stateless variant of SCAFFOLD, defined in Algorithm 8) at step k use the update rule:

$$x_{i,k}^{(s,r)} \leftarrow x_{i,k-1}^{(s,r)} - \eta_l^{(s,r)}(\tilde{\nabla}F_i(x_{i,k-1}^{(s,r)}) + \tilde{\nabla}F(x^{(s,r)}) - \tilde{\nabla}F(x^{(s,r)}))$$

Global round update: All the clients send back their effective updates to the server. Then the server takes an average over N effective updates to compute the pseudo-gradient $G^{(s,r)}$. The local updates and the collection process are abstracted by the step $G^{(s,r)} \leftarrow \mathsf{ClientOPT}(\eta_l^{(s,r)}, x^{(s,r)})$, where

$$\mathsf{ClientOPT}(\eta_l^{(s,r)}, x^{(s,r)}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\tilde{\nabla}F_i(x_{i,k-1}^{(s,r)}).$$

For (Accelerated) SGD (Algorithms 2 and 3) we set $\eta_l^{(s,r)} = 0$, and so $x_{i,k}^{(s,r)} = x^{(s,r)}$. Therefore, we can simplify $\mathsf{ClientOPT}$ as a simple re-scaled (multiplied by K) minibatched stochastic gradient of the global function, with a batchsize of K, i.e.

$$\mathsf{ClientOPT}(\eta_l^{(s,r)} = 0, x^{(s,r)}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\tilde{\nabla}F_i(x^{(s,r)})$$

Finally, the server takes a (accelerated) stochastic pseudo-gradient descent step using its round iterate history, and the computed pseudo-gradient $G^{(s,r)}$ and global stepsize $\eta_g^{(s,r)}$.

Stage update: Once R rounds are complete, the server completes the stage by taking a weighted average of its global iterates with weights $\{\{\{w_{i,k}^{(s,r)}\}_{i\le N}\}_{k\le K}\}_{r\le R}$:

$$\mathsf{StageWeightedAverage}(\{\{w_k^{(s,r)}\}_{k\le K}\}_{r\le R}) = \frac{1}{W^{(s)}}\sum_{r=1}^{R}\sum_{k=1}^{K}w_k^{(s,r)}(\frac{1}{N}\sum_{i=1}^{N}x_{i,k}^{(s,r)}),$$

where $W^{(s)} = \sum_{r=1}^{R}\sum_{k=1}^{K}w_k^{(s,r)}$. Although this weighted sum is commonly used to facilitate theoretical analysis (Karimireddy et al., 2020b; Gorbunov et al., 2020), it is often unnecessary in practice (i.e., we can just output the last iterate). The weights are determined by strong convexity μ and stepsizes, and do not require further tuning.

3.1 TWO STAGE ALGORITHMS

We begin with a two-stage instantiation of Algorithm 1. The main idea is to first run FedAvg for a constant fraction of the round budget R, which reduces error to the "heterogeneity floor". Then we use a minibatch algorithm to complete convergence.

Intuition: Fig. 1 shows a toy example that conveys the main intuition behind this idea. Consider two strongly convex client objectives: $F_1(x) = \frac{1}{2}(x-1)^2$ and $F_2(x) = (x+1)^2$. The global objective $F(x) = \frac{F_1(x)+F_2(x)}{2}$ is their average. Fig. 1 (top) plots the objectives, and Fig. 1 (bottom) displays their gradients. Far from the optimum, due to strong convexity, all client gradients point towards the optimal solution; specifically, this occurs when $x \in (-\infty, -1] \cup [1, \infty)$. In this regime, clients can use local steps (e.g., FedAvg) to save on communication while still ensuring that per-client local updates are consistent with each other. On the other hand, close to the optimum, i.e., when $x \in (-1, 1)$, some client gradients may point away from the optimum. This suggests that clients should not take local steps to avoid driving the global solution away from the optimum. Therefore, when we are close to the optimum (where "closeness" is determined by the clients' data heterogeneity), we use an algorithm without local steps (e.g. SGD), since such algorithms are less affected by client gradient disagreement. This intuition holds more generally for other problems because we have assumed strong convexity.

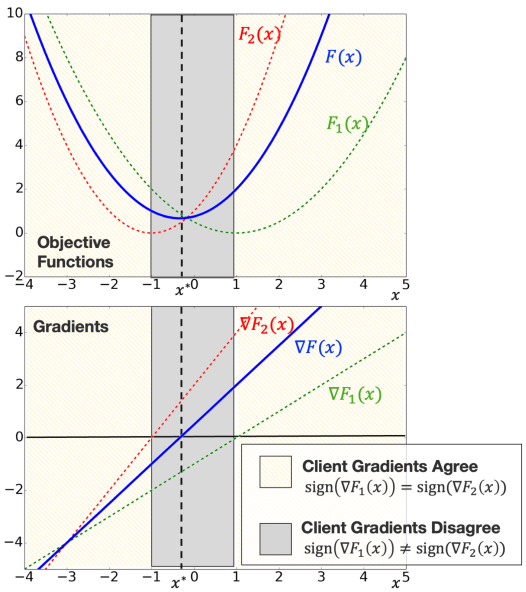


Figure 1: This toy example gives the intuition behind our multistage algorithm. We have two clients objectives: $F_1(x)$ and $F_2(x)$. $F(x)$ the global objective is their average. The top plot displays the objectives and the bottom plots displays the gradients. In the regions where client gradients agree in direction, i.e. $(-\infty, -1] \cup [1, \infty)$ it might be better to use an algorithm with local steps (like FedAvg), and in the region where the gradient disagree in direction, i.e. $(-1, 1)$ it may be better to use an algorithm without local steps (like SGD).

Our observation that client gradients will disagree as one approaches the optimal solution has been observed in practice even for objective functions that are not convex. Charles et al. (2021) observe that over the course of FedAvg execution on neural network StackOverflow next word prediction, client gradients become more orthogonal from each other, i.e. client gradients have different directions.

Our technique parallels multistage algorithms for obtaining optimal dependence on noise in convex optimization (Aybat et al., 2019). There, a large initial stepsize reduces the bias term (i.e. the exponential decay term) in the convergence rate; the stepsize is then decayed to reduce the noise term until convergence. It also has similarities to the approach of (Wang & Joshi, 2019), which reduces the number of local steps during training.

In this subsection, we find that two stages suffice to achieve low round complexities if $\Delta, \zeta^2, \sigma^2$ are known. In the next subsection, we show how to achieve low round complexities when $\Delta, \zeta^2, \sigma^2$ are unknown, by using more than two stages. We first show how to combine FedAvg with SGD to obtain exponential gains in round complexity compared to either baseline.

Theorem 1 (Informal version of Thm. 4 in Appendix)**.** *If we run Algorithm 1 over $S = 2$ stages with large enough number of local rounds K, such that (i) the first stage runs FedAvg as for $R/2$ rounds under some parameter settings, and (ii) the second stage runs SGD ($\gamma^{(2)} = 0$) for $R/2$ rounds under some other parameter settings, then output of this second stage $x^{(2)}$ satisfies*

$$\mathbb{E}F(x^{(2)}) - F(x^*) \leq \tilde{O}((\frac{\beta^2\zeta^2}{\mu^3 R^2})\exp(-\frac{R}{\kappa}) + \frac{\sigma^2}{\mu NKR}) \tag{8}$$

(Proof in App. D) This algorithm already exponentially improves the dependence of the round complexity on heterogeneity from the previous state of the art of $\frac{\beta\zeta^2}{\mu^2 R^2}$ to $(\frac{\beta^2\zeta^2}{\mu^3 R^2})\exp(-\frac{R}{\kappa})$.

Next, we obtain our main result, which tightens the dependency on κ to (nearly, up to exponentially shrinking condition number factors and supposing ζ_*^2 is of the order of ζ^2) match the lower bound in Eq. (6) by replacing SGD with ASG.

Theorem 2 (Informal version of Thm. 5 in Appendix)**.** *If we run Algorithm 1 over $S = 2$ stages with large enough number of local rounds K, such that (i) the first stage runs FedAvg as for $R/2$ rounds under some parameter settings, and (ii) the second stage runs ASG for $R/2$ rounds under some other parameter settings, then output of this second stage $x^{(2)}$ satisfies*

$$\mathbb{E}F(x^{(2)}) - F(x^*) \leq \tilde{\mathcal{O}}(\frac{\beta\zeta^2}{\mu^2 R^2}\exp(-\frac{R}{\sqrt{\kappa}}) + \frac{\sigma^2}{\mu NKR} + \frac{\beta\sigma^2}{\mu^2 NKR^3}). \tag{9}$$

(Proof in Section E). Observe that this rate is strictly better than the FedAvg rate (Eq. (3)) and is also strictly better than the AC-SA rate (Eq. (4)) when $\frac{\beta\zeta^2}{\mu^2 R^2} \leq \Delta$ (since the AC-SA rate requires that $R \geq \sqrt{\kappa}$). In practice, the initialization can be far from the optimum, leading to large Δ. The same rates can be achieved when using SS-Local-SGD instead of FedAvg as the federated optimization algorithm in the first stage of both of the above two-stage algorithms.

3.2 Multistage algorithms

The previous algorithms require knowledge of several data- and problem-dependent parameters, such as β, μ, σ^2, ζ^2, and Δ. Although the FL literature commonly assumes knowledge of such parameters (Woodworth et al., 2020a; Karimireddy et al., 2020b; Gorbunov et al., 2020), this may be unrealistic in practice. In this section, we explore how to use multistage algorithms to remove dependencies on all such parameters except β, which can be estimated from line search techniques (Beck & Teboulle, 2009; Schmidt et al., 2015), and μ, which can be estimated from regularization (which is often used in machine learning problems). The key idea is to divide our total stages S into two *superstages*: the first superstage uses one optimization algorithm (e.g., FedAvg, or SS-Local-SGD) with one hyperparameter schedule, and the second superstage uses another (e.g., SGD, or ASG).

Theorem 3 (Informal version of Theorem 7 in Appendix)**.** *Let Algorithm 1 run over two superstages for a total of R rounds over S stages, with $R \geq 4\sqrt{\kappa}$. In the first superstage, run Multistage FedAvg (M-FedAvg) (see App. H) for $R/2$ rounds (Thm. 8). In the second superstage, run Multistage Accelerated Stochastic Gradient (M-ASG) (Aybat et al., 2019) with some other parameter choices and batch-size B. Then for large enough B the output $x^{(S)}$ satisfies*

$$\mathbb{E}F(x^{(S)}) - F(x^*) \leq \mathcal{O}((\kappa\Delta\exp(-R) + \frac{\beta^2\zeta^2}{\mu^3 R^2})\exp(-\frac{R}{\sqrt{\kappa}}) + \frac{\sigma^2}{\mu BNKR}).$$

(Proof in App. G) For simplicity, suppose $\zeta^2 > 0$, $R \geq \log(\frac{\mu^4\Delta}{\beta^3\zeta^2})$ (where R is large enough for $\kappa\Delta\exp(-R)$ to match $\frac{\beta^2\zeta^2}{\mu^3 R^2}$; this is a relatively few number of rounds). Then this multistage algorithm has a better convergence rate with respect to communication cost than both FedAvg and AC-SA when $0 < \zeta^2 < \frac{\mu^2\Delta}{\beta}$(note that now the variance is $\frac{\sigma^2}{B}$ and that both algorithms require $R \geq \Omega(\sqrt{\kappa})$). This is done without knowledge of ζ^2, σ^2, or Δ. However, note we do not achieve the optimal rate like in Thm. 2 due to the additional term of $\kappa\Delta\exp(-R)$.

4 Experiments

We evaluate the utility of our framework on strongly convex and nonconvex settings. We compare the communication round complexities of four single-stage baselines: FedAvg, SGD, ASG, and SCAFFOLD. We compare these baselines to both two-stage variants and multistage variants.

Convex Optimization (Logistic Regression) We first study federated regularized logistic regression, which is strongly convex (objective function in App. K.1). In this experiment, we use the

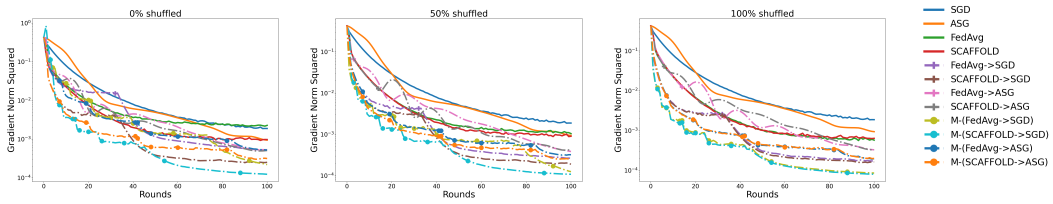
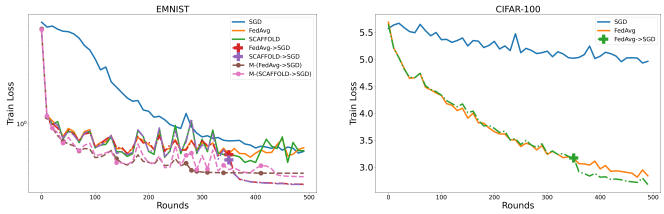


Figure 2: Stochastic gradients are computed with 1% of the data. Each datapoint is the average of 1000 runs. Plot titles denote data homogeneity (§ 4). "X→Y" denotes a multistage algorithm with X as the first superstage and Y as the second superstage. Across all heterogeneity levels, the multistage algorithms perform the best. Markers denote when a stage transition occurs.



Algorithm	Test Accuracy
SGD	0.0807
FedAvg	0.3175
FedAvg → SGD	**0.3360**

Figure 3: **Left:** EMNIST training loss, **Middle:** CIFAR-100 training loss, **Right:** CIFAR-100 test accuracy (↑). In all plots, markers denote stage transitions. In EMNIST experiments, we observe that two-stage algorithms have the fastest convergence in communication rounds.

MNIST dataset of handwritten digits (LeCun et al., 2010). We (roughly) control client heterogeneity by creating "clients" by shuffling samples from different digit classes. The details of this shuffling process are described in App. K.1; if a dataset is more shuffled (more homogeneous), it roughly corresponds to a lower heterogeneity dataset.

Fig. 2 compares the convergence of multi-stage algorithms to their single-stage counterparts in the stochastic gradient setting (minibatches are 1% of a client's data), over $R = 100$ rounds. We observe that in all heterogeneity levels, multistage algorithms (whether two or more stages) outperform other baselines. We also observe that SCAFFOLD→SGD outperforms all other curves, including SCAFFOLD→ASG. This can be explained by the fact that acceleration introduces additional noise terms in the stochastic setting as shown in Thm. 5, which can contribute to error if one does not take K large enough (we set $K = 20$ in the convex experiments).

Nonconvex Optimization (Neural Networks) We also evaluated nonconvex image classification tasks with convolutional neural networks. We started with digit classification over EMNIST (Cohen et al., 2017) (details in App. K.3), where handwritten characters are partitioned by author. Figure 3 (**Left**) displays the training losses. The multistage ($S > 2$) algorithms do not perform as well as two-stage algorithms, possibly because we use the same stepsize schedule for both superstages to keep the number of hyperparameters low. While successful in the convex setting, this approach does not work as well in our nonconvex experiments, possibly due to increased task difficulty. We also considered image classification with ResNet-18 on CIFAR-100 (Krizhevsky, 2009) (details in App. K.4). Figure 3 (**Middle**) displays the training losses on CIFAR-100; SCAFFOLD experiments are not included due to memory constraints. The two-stage algorithms again perform the best. In Fig. 3 (**Right**), we display the test accuracies on CIFAR-100; two-stage FedAvg → SGD achieves the highest test accuracy (test accuracies for EMNIST in App. L.2).

5 Conclusion

In this paper, we show that multistage parameter schedules improve FL communication complexity in theory and in practice. In App. A, we discuss why our theoretical results may help explain the benefits of multistaging in practice. Future work includes tightening our upper bounds from ζ^2 to ζ_*^2 and getting rid of the additional $\kappa\Delta\exp(-R)$ term in Thm. 7.

REFERENCES

Maruan Al-Shedivat, Jennifer Gillenwater, Eric Xing, and Afshin Rostamizadeh. Federated learning via posterior averaging: A new perspective and practical algorithms. *arXiv preprint arXiv:2010.05273*, 2020.

Necdet Serhat Aybat, Alireza Fallah, Mert Gurbuzbalaban, and Asuman Ozdaglar. A universally optimal multistage accelerated stochastic gradient method. *arXiv preprint arXiv:1901.08022*, 2019.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Zachary Charles and Jakub Konečnỳ. On the outsized importance of learning rates in local update methods. *arXiv preprint arXiv:2007.00878*, 2020.

Zachary Charles, Zachary Garrett, Zhouyuan Huo, Sergei Shmulyian, and Virginia Smith. On large-cohort training for federated learning. *arXiv preprint arXiv:2106.07820*, 2021.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.

Alireza Fallah, Asuman Ozdaglar, and Sarath Pattathil. An optimal multistage stochastic gradient method for minimax problems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 3573–3579. IEEE, 2020.

Saeed Ghadimi and Guanghui Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization i: A generic algorithmic framework. *SIAM Journal on Optimization*, 22(4):1469–1492, 2012.

Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. Local sgd: Unified theory and new efficient methods. *arXiv preprint arXiv:2011.02828*, 2020.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020a.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020b.

Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pp. 4519–4529. PMLR, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. A unified theory of decentralized sgd with changing topology and local updates. In *International Conference on Machine Learning*, pp. 5381–5393. PMLR, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smithy. Feddane: A federated newton-type method. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pp. 1227–1231. IEEE, 2019.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Aritra Mitra, Rayana Jaafar, George J. Pappas, and Hamed Hassani. Linear Convergence in Federated Learning: Tackling Client Heterogeneity and Sparse Gradients. *arXiv e-prints*, art. arXiv:2102.07053, February 2021.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Mark Schmidt, Reza Babanezhad, Mohamed Ahmed, Aaron Defazio, Ann Clifton, and Anoop Sarkar. Non-uniform stochastic average gradient method for training conditional random fields. In *artificial intelligence and statistics*, pp. 819–828. PMLR, 2015.

Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

Jianyu Wang and Gauri Joshi. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

Jianyu Wang and Gauri Joshi. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd. In *SysML*, 2019.

Jianyu Wang, Anit Kumar Sahu, Zhouyi Yang, Gauri Joshi, and Soummya Kar. Matcha: Speeding up decentralized sgd via matching decomposition sampling. In *2019 Sixth Indian Control Conference (ICC)*, pp. 299–300. IEEE, 2019a.

Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. Slowmo: Improving communication-efficient distributed sgd with slow momentum. *arXiv preprint arXiv:1910.00643*, 2019b.

Blake Woodworth, Kumar Kshitij Patel, and Nathan Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *arXiv preprint arXiv:2006.04735*, 2020a.

Blake Woodworth, Kumar Kshitij Patel, Sebastian Stich, Zhen Dai, Brian Bullins, Brendan Mcmahan, Ohad Shamir, and Nathan Srebro. Is local sgd better than minibatch sgd? In *International Conference on Machine Learning*, pp. 10334–10343. PMLR, 2020b.

Blake Woodworth, Brian Bullins, Ohad Shamir, and Nathan Srebro. The min-max complexity of distributed stochastic convex optimization with intermittent communication. *arXiv preprint arXiv:2102.01583*, 2021.

A DISCUSSION: STEPSIZE DECAY IN PRACTICE

In this section, we make the case that our strategy of switching from a local step-utilizing algorithm (like FedAvg) to a centralized algorithm (like SGD) is similar to the stepsize decay strategies in (Reddi et al., 2020; Charles & Konečnỳ, 2020), thus potentially explaining the empirical gains they see in their papers. In (Charles & Konečnỳ, 2020), the decay scheme is roughly as follows: if the loss decreases too little, then reduce the outer stepsize by a factor of 0.9 and the inner stepsize by a factor of 0.1. It is easy to see that in a relatively few number of decay events, their algorithm will essentially have switched from a local step-utilizing algorithm to a centralized algorithm. Therefore, it is plausible that the practical improvements they observe from their decay scheme can largely be explained by our results.

In (Reddi et al., 2020), the definition of inner/outer stepsize is a bit different to ours or (Charles & Konečnỳ, 2020). They define the inner stepsize η_l as the stepsize that updates

$$x_{i,k} \leftarrow x_{i,k-1} - \eta_l \tilde{\nabla} F_i(x_{i,k-1}) \tag{10}$$

and then the clients send $\delta_i^r := x_{i,K} - x_{i,0}$ to the server, where the server averages $\delta^r := \frac{1}{N}\sum_{i=1}^N \delta_i^r$ and then updates the global model with outer stepsize η_g as

$$x^r \leftarrow x^{r-1} + \eta_g \delta^r \tag{11}$$

where we leave out the modifications for adaptivity that they introduce for simplicity (where adaptivity is in the coordinate-wise sense like in Adam (Kingma & Ba, 2014)). In their case, they decay their η_l with a factor of 0.1 every 500 rounds. In this case as well, we see that the algorithm will switch from a local step-utilizing algorithm to a centralized algorithm with a few decay events. So we may also plausibly conclude that the practical improvements they see from stepsize decay can largely be explained by our results.

B ALGORITHM PSEUDOCODE

Algorithm 2 SGD

Input: Minibatch size K, rounds R, stepsize η, initial point $x^{(0)}$, number of devices N
for $r = 0, \ldots, R-1$ **do**
  Server sends $x^{(r)}$ to all clients
  ▷ **Run a step of SGD**
  for client $i = 1, \ldots, N$ *in parallel* **do**
    Sample K datapoints $\{z_{i,k}\}_{1 \le k \le K}$
    Client **communicates** $\sum_{k=1}^K \nabla f(x^{(r)}; z_{i,k})$ to server
  end for
  $x^{(r+1)} \leftarrow x^{(r)} - \frac{\eta}{NK}\sum_{i=1}^N \sum_{k=1}^K \nabla f(x^{(r)}; z_{i,k})$
end for
return $x^{(R)}$

B.1 ALGORITHM 2 IN TERMS OF ALGORITHM 1

Set $\eta_g^{(1,r)} = \eta/K$ $\eta_l^{(1,r)} = 0$, $S = 1$, $\gamma^{(1)} = 0$, and $R^{(1)} = R$. For StageWeightedAverage simply return the last iterate $x^{(1,R^{(1)})}$.

Algorithm 3 ASG

Input: Minibatch size K, rounds R, stepsize η, initial point $x^{(0)} = x^{(1)}$, strong convexity μ number of devices N
for $r = 0, \ldots, R-1$ **do**
  Server sends $x^{(r)}$ to all clients
  ▷ **Run a step of ASG**
  for client $i = 1, \ldots, N$ *in parallel* **do**
    Sample K datapoints $\{z_{i,k}\}_{1 \leq k \leq K}$
    Client **communicates** $\sum_{k=1}^{K} \nabla f(x^{(r)}; z_{i,k})$ to server
  end for
  $\gamma^{(r)} \leftarrow \frac{1-\sqrt{\eta\mu}}{1+\sqrt{\eta\mu}}$
  $y^{(r)} \leftarrow (1+\gamma^{(r)})x^{(r)} - \gamma^{(r)}x^{(r-1)}$
  $x^{(r+1)} \leftarrow y^{(r)} - \frac{\eta}{NK}\sum_{i=1}^{N}\sum_{k=1}^{K} \nabla f(x^{(r)}; z_{i,k})$
end for
return $x^{(R)}$

B.2 Algorithm 3 in terms of Algorithm 1

Set $\eta_g^{(1,r)} = \eta/K$ $\eta_l^{(1,r)} = 0$, $S = 1$, $\gamma^{(1)} = \frac{1-\sqrt{\eta\mu}}{1+\sqrt{\eta\mu}}$ where μ is strong convexity, and $R^{(1)} = R$. For StageWeightedAverage simply return the last iterate $x^{(1,R^{(1)})}$.

Algorithm 4 M-SGD

Input: Minibatch size K, round sequence $\{R^{(s)}\}_{s \leq S}$, stepsize schedule $\{\eta^{(s)}\}_{s \leq S}$, initial point $x^{(0)}$, number of devices N
for $s = 1, \ldots, S$ **do**
  for $r = 0, \ldots, R^{(s)} - 1$ **do**
    Server sends $x^{(s,r)}$ to all clients
    ▷ **Run a step of SGD**
    for client $i = 1, \ldots, N$ *in parallel* **do**
      Sample K datapoints $\{z_{i,k}\}_{1 \leq k \leq K}$
      Client **communicates** $\sum_{k=1}^{K} \nabla f(x^{(s,r)}; z_{i,k})$ to server
    end for
    $x^{(s,r+1)} \leftarrow x^{(s,r)} - \frac{\eta^{(s)}}{NK}\sum_{i=1}^{N}\sum_{k=1}^{K} \nabla f(x^{(s,r)}; z_{i,k})$
  end for
end for
return $x^{(S,R^{(S)})}$

B.3 Algorithm 4 in terms of Algorithm 1

Set $\eta_g^{(s,r)} = \eta^{(s)}/K$ $\eta_l^{(1,r)} = 0$, same number of stages S, $\gamma^{(s)} = 0$, and same round sequence $R^{(s)}$. For StageWeightedAverage simply return the last iterate $x^{(S,R^{(S)})}$.

B.4 Algorithm 5 in terms of Algorithm 1

Set $\eta_g^{(s,r)} = \eta^{(s)}/K$ $\eta_l^{(1,r)} = 0$, same number of stages S, $\gamma^{(s)} = \frac{1-\sqrt{\eta^{(s)}\mu}}{1+\sqrt{\eta^{(s)}\mu}}$ where μ is strong convexity, and same round sequence $R^{(s)}$. For StageWeightedAverage simply return the last iterate $x^{(S,R^{(S)})}$.

Algorithm 5 M-ASG

Input: Minibatch size K, round sequence $\{R^{(s)}\}_{s\leq S}$, stepsize schedule $\{\eta^{(s)}\}_{s\leq S}$, initial point $x^{(0)} = x^{(1)}$, strong convexity μ number of devices N
for $s = 1, \ldots, S$ **do**
  for $r = 1, \ldots, R^{(s)}$ **do**
    Server sends $x^{(s,r)}$ to all clients
    ▷ **Run a step of ASG**
    for client $i = 1, \ldots, N$ *in parallel* **do**
      Sample K datapoints $\{z_{i,k}\}_{1\leq k\leq K}$
      Client **communicates** $\sum_{k=1}^{K} \nabla f(x^{(s,r)}; z_{i,k})$ to server
    end for
    $\gamma^{(s)} \leftarrow \frac{1-\sqrt{\eta^{(s)}\mu}}{1+\sqrt{\eta^{(s)}\mu}}$
    $y^{(s,r)} \leftarrow (1+\gamma^{(s,r)})x^{(s,r)} - \gamma^{(s,r)}x^{(s,r-1)}$
    $x^{(s,r+1)} \leftarrow y^{(s,r)} - \frac{\eta^{(s)}}{NK}\sum_{i=1}^{N}\sum_{k=1}^{K} \nabla f(x^{(s,r)}; z_{i,k})$
  end for
end for
return $x^{(S,R^{(s)})}$

B.5 Algorithm 6 in terms of Algorithm 1

Set $\eta_g^{(s,r)} = \eta_l^{(s,r)} = \eta$, $S = 0$, $\gamma^{(s,r)} = 0$, $R^{(1)} = R$. $x_{i,k}^{(s,r)} = x_{i,k}^{(r)}$. $w_k^{(1,r)} = (1-\mu\eta)^{rK+k+1}$. For ClientOPT,

$$\mathsf{ClientOPT}(\eta_l^{(s,r)}, x^{(s,r)}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} \tilde{\nabla} F_i(x_{i,k-1}^{(s,r)}).$$

where the $\tilde{\nabla} F_i(x_{i,k-1}^{(s,r)})$'s come from

$$x_{i,k}^{(s,r)} \leftarrow x_{i,k-1}^{(s,r)} - \eta_l^{(s,r)}\tilde{\nabla} F_i(x_{i,k-1}^{(s,r)}) \tag{12}$$

Algorithm 6 FedAvg

Input: Number of local steps K, rounds R, stepsize η_l, η_g, initial point $x^{(0)}$, number of devices N
for $r = 0, \ldots, R-1$ **do**
  Server **communicates** $x^{(r)}$ to all clients
  for client $i = 1, \ldots, N$ *in parallel* **do**
    $x_{i,0}^{(r)} \leftarrow x^{(r)}$
    for $k = 1, \ldots, K$ **do**
      Randomly select sample $z_{i,k-1}$
      $x_{i,k}^{(r)} \leftarrow x_{i,k-1}^{(r)} - \eta \cdot \nabla f(x_{i,k-1}^{(r)}; z_{i,k-1})$
    end for
    $c_i \leftarrow \frac{1}{K}\sum_{k=1}^{K} \nabla f(x_{i,k-1}^{(r)}; z_{i,k-1})$
    Client **communicates** $\sum_{k=1}^{K} \nabla f(x_{i,k-1}^{(r)}; z_{i,k-1})$ to server
  end for
  ▷ **Perform server update**
  $x^{(r+1)} \leftarrow x^{(r)} - \frac{\eta}{N}\sum_{i\in[N]}\sum_{k=1}^{K} \nabla f(x_{i,k-1}^{(r)}; z_{i,k-1})$
end for
return $x^{(R)}$

B.6 ALGORITHM 7 IN TERMS OF ALGORITHM 1

Set $\eta_g^{(s,r)} = \eta_l^{(s,r)} = \eta$, $S = 0$, $\gamma^{(s,r)} = 0$, $R^{(1)} = R$. $w_k^{(1,r)} = (1-\mu\eta)^{rK+k+1}$. $x_{i,k}^{(s,r)} = x_{i,k}^{(r)}$. For ClientOPT,

$$\mathsf{ClientOPT}(\eta_l^{(s,r)}, x^{(s,r)}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\tilde{\nabla}F_i(x_{i,k-1}^{(s,r)}).$$

where the $\tilde{\nabla}F_i(x_{i,k-1}^{(s,r)})$'s come from

$$x_{i,k}^{(s,r)} \leftarrow x_{i,k-1}^{(s,r)} - \eta_l^{(s,r)}(\tilde{\nabla}F_i(x_{i,k-1}^{(s,r)} + c - c_i)) \tag{13}$$

where $c_i = \frac{1}{K}\sum_{i=1}^{K}\tilde{\nabla}F_i(x_{i,k}^{(s,r-1)})$ if the previous round is in the same stage and $c_i = \frac{1}{K}\sum_{i=1}^{K}\tilde{\nabla}F_i(x_{i,k}^{(s-1,R^{(s-1)})})$ if the previous round is in an earlier stage. $c = \frac{1}{N}\sum_{i=1}^{N} c_i$. In Algorithm 1 we do not specify states (for c and c_i), though one can easily include them.

Algorithm 7 SCAFFOLD

Input: Number of local steps K, rounds R, stepsize η_l, η_g, initial point $x^{(0)}$, control variates c and c_i, number of devices N
for $r = 0, \ldots, R-1$ **do**
  Server **communicates** $x^{(r)}$ to all clients
  Server **communicates** c to all clients
  for client $i = 1, \ldots, N$ *in parallel* **do**
    $x_{i,0}^{(r)} \leftarrow x^{(r)}$
    for $k = 1, \ldots, K$ **do**
      Randomly select sample $z_{i,k-1}$
      $x_{i,k}^{(r)} \leftarrow x_{i,k-1}^{(r)} - \eta_l \cdot (\nabla f(x_{i,k-1}^{(r)}; z_{i,k-1}) + c - c_i)$
    end for
    $c_i \leftarrow \frac{1}{K}\sum_{k=0}^{K-1}\nabla f(x_{i,k-1}^{(r)}; z_{i,j})$
    Client **communicates** $\sum_{k=0}^{K-1}\nabla f(x_{i,k-1}^{(r)}; z_{i,k})$ to server
    Client **communicates** c_i to server
  end for
  ▷ **Perform server update**
  $x^{(r+1)} \leftarrow x^{(r)} - \frac{\eta_g}{N}\sum_{i\in[N]}\sum_{k=0}^{K-1}\nabla f(x_{i,k-1}^{(r)}; z_{i,k})$
  $c \leftarrow \frac{1}{N}\sum_{i=1}^{N} c_i$
end for
return $x^{(R)}$

Algorithm 8 SCAFFOLD (SS-Local-SGD)

Input: Number of local steps K, rounds R, stepsize η_l, η_g, initial point $x^{(0)}$, number of devices N
for $r = 0, \ldots, R-1$ **do**
  Server **communicates** $x^{(r)}$ to all clients
  For client $i \in [n]$ compute $c_i \leftarrow \frac{1}{K}\sum_{i=1}^{K} \nabla f(x^{(r)}; z_{i,k})$ for randomly drawn $z_{i,k}$
  Clients **communicate** c_i's to server
  $c \leftarrow \frac{1}{N}\sum_{i=1}^{N} c_i$
  Server **communicates** c to clients
  for client $i = 1, \ldots, N$ *in parallel* **do**
    $x_{i,0}^{(r)} \leftarrow x^{(r)}$
    for $k = 1, \ldots, K$ **do**
      Randomly select sample $z_{i,k-1}$
      $x_{i,k}^{(r)} \leftarrow x_{i,k-1}^{(r)} - \eta \cdot (\nabla f(x_{i,k-1}^{(r)}; z_{i,k-1}) + c - c_i)$
    end for
    $c_i \leftarrow \frac{1}{K}\sum_{k=0}^{K-1} \nabla f(x_{i,k-1}^{(r)}; z_{i,j})$
    Client **communicates** $\sum_{k=0}^{K-1} \nabla f(x_{i,k-1}^{(r)}; z_{i,k})$ to server
  end for
  ▷ **Perform server update**
  $x^{(r+1)} \leftarrow x^{(r)} - \frac{\eta}{N}\sum_{i\in[N]}\sum_{k=0}^{K-1} \nabla f(x_{i,k-1}^{(r)}; z_{i,k})$
end for
return $x^{(R)}$

B.7 Algorithm 8 in terms of Algorithm 1

Set $\eta_g^{(s,r)} = \eta_l^{(s,r)} = \eta$, $S = 0$, $\gamma^{(s,r)} = 0$, $R^{(1)} = R$. $x_{i,k}^{(s,r)} = x_{i,k}^{(r)}$. $w_k^{(1,r)} = (1-\mu\eta)^{rK+k+1}$. For ClientOPT,

$$\mathsf{ClientOPT}(\eta_l^{(s,r)}, x^{(s,r)}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} \tilde{\nabla} F_i(x_{i,k-1}^{(s,r)}).$$

where the $\tilde{\nabla} F_i(x_{i,k-1}^{(s,r)})$'s come from

$$x_{i,k}^{(s,r)} \leftarrow x_{i,k-1}^{(s,r)} - \eta_l^{(s,r)}(\tilde{\nabla} F_i(x_{i,k-1}^{(s,r)} + c - c_i)) \tag{14}$$

where $c_i = \frac{1}{K}\sum_{i=1}^{K} \tilde{\nabla} F_i(x^{(s,r)})$, $c = \frac{1}{N}\sum_{i=1}^{N} c_i$.

C Technicalities

In this section we collect all the assumptions we make in the paper for convenience.

C.1 Assumptions

Assumption 1. *F_i's are μ-strongly convex for $\mu > 0$, i.e.*

$$\langle \nabla F_i(x), y - x\rangle \le -(F_i(x) - F_i(y) + \frac{\mu}{2}\|x-y\|^2) \tag{15}$$

for any i, x, y.

Assumption 2. *F_i's are β-smooth where*

$$\|\nabla F_i(x) - \nabla F_i(y)\| \le \beta\|x-y\| \tag{16}$$

for any i, x, y.

Assumption 3. *F_i's have a uniform upper bound on the gradient dissimilarity (heterogeneity), defined as*

$$\zeta^2 := \sup_x \frac{1}{N}\sum_{i\in[N]} \|\nabla F(x) - \nabla F_i(x)\|^2, \tag{17}$$

Assumption 4. *F_i's have an upper bound on the gradient dissimilarity (heterogeneity) at the optimum, defined as*

$$\zeta_*^2 := (1/N) \sum_{i \in [N]} \|\nabla F_i(x^*)\|^2 \tag{18}$$

Assumption 5. *We assume that the F_i's have bounded variance of stochastic gradients, where z_i is a randomly drawn data point from client i's data:*

$$\mathbb{E}_{z_i} \|\nabla f(x; z_i) - \nabla F_i(x)\|^2 \leq \sigma^2. \tag{19}$$

Assumption 6. *We assume that the initial suboptimality gap is upper bounded by Δ, where x_0 is the initial feasible iterate and x^* is the global minimizer.*

$$F(x_0) - F(x^*) \leq \Delta \tag{20}$$

C.2 Definitions

Definition 1. *An algorithm is said to have converged, or to have satisfied the convergence criterion if*

$$\mathbb{E}F(\hat{x}) - F(x^*) \leq \epsilon \tag{21}$$

where ϵ is some user-chosen value.

Definition 2. *A rate of convergence of an algorithm in our setting is an upper bound on the error of the returned solution $\hat{x}$ in terms of N, σ^2, ζ, R, K, Δ, β, μ, ζ_*, e.g.*

$$\mathbb{E}F(\hat{x}) - F(x^*) \leq M(R, K, N, \Delta, \zeta, \zeta_*, \beta, \mu, \sigma^2) \tag{22}$$

where M is a function that returns a one-dimensional value.

D Proof of Thm. 1

Theorem 4 (Formal version of Thm. 1)**.** *Run Algorithm 1 over $S = 2$ stages. In the first stage, use FedAvg[3] as for $R/2$ rounds under the following parameter settings: $\eta = \eta_g^{(1,r)} = \eta_l^{(1,r)} = \mathcal{O}(\min\{\frac{1}{\beta}, \frac{1}{\mu K R}\})$, $S = 1$, $\gamma^{(1)} = 0$, and $w_k^{(1,r)} = (1 - \mu\eta)^{rK+k+1}$. In the second stage, run SGD (so $\gamma^{(2)} = 0$) for $R/2$ rounds under the following settings:*

1. *If $R > \frac{4\beta}{\mu}$ and $r < \lceil \frac{R}{2} \rceil$: $\eta_l^{(2,r)} = 0$, $\eta_g^{(2,r)} = \eta_g = \frac{1}{4\beta}$, $w_k^{(2,r)} = 0$*
2. *If $R > \frac{4\beta}{\mu}$ and $r \geq \lceil R/2 \rceil$: $\eta_l^{(2,r)} = 0$,$\eta_g^{(2,r)} = \eta_g = \frac{2}{\mu(\kappa - R - \lceil \frac{R}{2} \rceil)}$, $w_k^{(2,r)} = (\kappa + r - \lceil \frac{R}{2} \rceil)^2$*
3. *If $R \leq \frac{4\beta}{\mu}$: $\eta_g^{(2,r)} = \frac{1}{4\beta}$, $\eta_l^{(2,r)} = 0$, $w_k^{(2,r)} = (1 - \mu\eta_g)^{-(rK+k+1)}$.*

Denote the output of this second stage by $x^{(2)}$. For large enough number of local rounds K, the convergence rate of this two-stage algorithm is

$$\mathbb{E}F(x^{(2)}) - F(x^*) \leq \tilde{O}((\frac{\beta^2 \zeta^2}{\mu^3 R^2}) \exp(-\frac{R}{\kappa}) + \frac{\sigma^2}{\mu N K R}) \tag{23}$$

Proof. First, let the total budget for the number of rounds be $2R$. We run the two stages of the algorithm with R rounds.

In the first stage, we run Algorithm 1 with

$$\eta = \eta_g^{(s,r)} = \eta_l^{(s,r)} = \min\{\frac{1}{4\beta}, \frac{\log(\max\{2, \min\{\frac{\mathbb{E}\|x^{(0)} - x^*\|^2 n \mu^2 R^2 K^2}{\sigma^2}, \frac{\mathbb{E}\|x^{(0)} - x^*\|^2 \mu^2 R^3 K^3}{6\beta K(\sigma^2 + 2K\zeta^2))}\}\}}{\mu K R}$$

[3]This is the FedAvg detailed in (Gorbunov et al., 2020), which differs from usual FedAvg in that it returns a weighted average of the iterates rather than the final iterate. This is mostly a proof convenience, and in practice one would return the final iterate as we do in the experiments.

$S = 1$, $\gamma^{(s)} = 0$, $w_k^{(s,r)} = (1-\mu\eta)^{rK+k+1}$. Note this is just running FedAvg Gorbunov et al. (2020) with the above stepsizes. Therefore, from the proof of Gorbunov et al. (2020)(Eq.82), we know that we get the following convergence rate for the output of the algorithm

$$\mathbb{E}F(x^{(1)}) - F(x^*) \leq 8\beta \exp(-\frac{\kappa KR}{8})\mathbb{E}\|x^{(0)} - x^*\|^2 \tag{24}$$

$$+ \frac{2(\Phi+1)\sigma^2}{\mu NKR} + \frac{6\beta(\Phi^2+1)\sigma^2}{\mu^2 KR^2} + \frac{12\beta(\Phi^2+1)\zeta^2}{\mu^2 R^2} \tag{25}$$

where $\Phi = \log(\max\{2, \min\{\frac{\mathbb{E}\|x^{(0)}-x^*\|^2 n\mu^2 R^2 K^2}{\sigma^2}, \frac{\mathbb{E}\|x^{(0)}-x^*\|^2 \mu^2 R^3 K^3}{6\beta K(\sigma^2+2K\zeta^2))}\}\}$.

In the second stage, we run Algorithm 1 on $x^{(1)}$ with $\beta^{(s)} = 0$ and

1. If $R > \frac{4\beta}{\mu}$ and $r < \lceil \frac{R}{2} \rceil$: $\eta_l^{(2,r)} = 0$, $\eta_g^{(2,r)} = \eta_g = \frac{1}{4\beta}$, $w_k^{(2,r)} = 0$
2. If $R > \frac{4\beta}{\mu}$ and $r \geq \lceil R/2 \rceil$: $\eta_g^{(2,r)} = \eta_g = \frac{2}{\mu(\kappa - R - \lceil \frac{R}{2} \rceil)}$, $w_k^{(2,r)} = (1-\mu\eta_g)^{-(rK+k+1)} = (\kappa + r - \lceil \frac{R}{2} \rceil)^2$
3. If $R \leq \frac{4\beta}{\mu}$: $\eta_g^{(2,r)} = \frac{1}{4\beta}$, $w_k^{(2,r)} = (1-\mu\eta_g)^{-(rK+k+1)}$.

Note that this is running SGD with the above stepsize schedule. From the proof from (Woodworth et al., 2020a), the output of this algorithm is

$$\mathbb{E}F(x^{(2)}) - F(x^*) \leq 128\kappa(\mathbb{E}F(x^{(1)}) - F(x^*)) \exp(-\frac{R}{8\kappa}) + \frac{72\sigma^2}{\mu NKR} \tag{26}$$

Plugging in (Eq. (24)), we get the claim. □

E PROOF OF THM. 2

Theorem 5 (Formal version of Thm. 2)**.** *Run Algorithm 1 over $S = 2$ stages. In the first stage, use FedAvg for for $R/2$ rounds under the parameter settings of Stage 1 from Thm. 4.*

In the second stage, run ASG for $R/2$ rounds with the following parameter settings:
$\eta_l^{(2,r)} = 0$, $\quad \eta_g^{(2,r)} = \eta_g = \mathcal{O}(\min\{\frac{1}{K\beta}, \frac{1}{\mu KR^2}\})$ $\quad \gamma^{(2)} = \frac{1-\sqrt{\mu\eta_g K}}{1+\sqrt{\mu\eta_g K}}$
$w_k^{(1,r)} = w_k^{(2,r)} = 0$ *for all k, r except for $w_1^{(2,R/2)} = 1$ (i.e. return the last iterate).*

Denote the output of this second stage by $x^{(2)}$. For large enough number of local rounds K, the expected convergence rate of this two-stage algorithm is

$$\mathbb{E}F(x^{(2)}) - F(x^*) \leq \tilde{\mathcal{O}}(\frac{\beta\zeta^2}{\mu^2 R^2}\exp(-\frac{R}{\sqrt{\kappa}}) + \frac{\sigma^2}{\mu NKR} + \frac{\beta\sigma^2}{\mu^2 NKR^3}). \tag{27}$$

Proof. First, let the total budget for the number of rounds be $2R$. We run the two stages of the algorithm with R rounds.

In the first stage, we run Algorithm 1 with

$$\eta = \eta_g^{(s,r)} = \eta_l^{(s,r)} = \min\{\frac{1}{4\beta}, \frac{\log(\max\{2, \min\{\frac{\mathbb{E}\|x^{(0)}-x^*\|^2 n\mu^2 R^2 K^2}{\sigma^2}, \frac{\mathbb{E}\|x^{(0)}-x^*\|^2 \mu^2 R^3 K^3}{6\beta K(\sigma^2+2K\zeta^2))}\}\}}{\mu KR}$$

$S = 1$, $\gamma^{(s)} = 0$, $w_k^{(s,r)} = (1-\mu\eta)^{rK+k+1}$. Note this is just running FedAvg (Gorbunov et al., 2020) with the above stepsizes. Therefore, from the proof of (Gorbunov et al., 2020), we know that we get the following convergence rate for the output of the algorithm

$$\mathbb{E}F(x^{(1)}) - F(x^*) \leq 8\beta \exp(-\frac{\kappa KR}{8})\mathbb{E}\|x^{(0)} - x^*\|^2 \tag{28}$$

$$+ \frac{2(\Phi+1)\sigma^2}{\mu NKR} + \frac{6\beta(\Phi^2+1)\sigma^2}{\mu^2 KR^2} + \frac{12\beta(\Phi^2+1)\zeta^2}{\mu^2 R^2} \tag{29}$$

where $\Phi = \log(\max\{2, \min\{\frac{\mathbb{E}\|x^{(0)}-x^*\|^2 n\mu^2 R^2 K^2}{\sigma^2}, \frac{\mathbb{E}\|x^{(0)}-x^*\|^2 \mu^2 R^3 K^3}{6\beta K(\sigma^2+2K\zeta^2))}\}\}$.

Next we run Algorithm 1 on $x^{(1)}$ with $\eta_l^{(s,r)} = 0$ and

$$\eta_g^{(s,r)} = \eta_g = \min\{\frac{1}{K\beta}, \frac{\log(\max\{2, \min\{\frac{2\Delta_1 N\mu KR}{\sigma^2}, \frac{2\Delta_1 \mu^2 NKR^3}{\beta\sigma^2}\})\}}{\mu KR^2}\} \tag{30}$$

$\gamma^{(s)} = \frac{1-\sqrt{\mu\eta_g K}}{1+\sqrt{\mu\eta_g K}}$, $S = 1$, and $w_k^{(s,r)} = 0$ for all k, r, s except for $w_1^{(1,R_1)} = 1$ and $\Delta_1 \leq 8\beta\exp(-\frac{\kappa R}{8})\mathbb{E}\|x^{(0)} - x^*\|^2 + \frac{2(\Phi+1)\sigma^2}{\mu NKR} + \frac{6\beta(\Phi^2+1)\sigma^2}{\mu^2 KR^2} + \frac{12\beta(\Phi^2+1)\zeta^2}{\mu^2 R^2}$. From Thm. 6 we know that

$$\mathbb{E}F(x^{(2)}) - F(x^*) \leq 2\Delta_1 \exp(-\frac{R}{\sqrt{\kappa}}) + \frac{\sigma^2(\Phi_2^{1/2}+1)}{\mu NKR} + \frac{\beta\sigma^2(\Phi_2^{3/2}+1)}{\mu^2 NKR^3} \tag{31}$$

where $\Phi_2 = \log(\max\{2, \min\{\frac{2\Delta_1\mu K}{\sigma^2}, \frac{2\Delta\mu^2 K^3}{\beta\sigma^2}\})\}$.

Plugging in Δ_1 gets us the desired conclusion. □

F THM. 6

Theorem 6. *Consider Nesterov's Accelerated Stochastic Gradient (ASG) method with the iteration sequence*

$$x_{k+1/2} = (1+\gamma)x_k - \gamma x_{k-1},\ x_{k+1} = x_{k+1/2} - \alpha\tilde{\nabla}F(x_{k+1/2}) \tag{32}$$

with $\gamma = \frac{1-\sqrt{\alpha\mu}}{1+\sqrt{\alpha\mu}}$ *and*

$$\alpha = \min\{\frac{1}{\beta}, \frac{\log(\max\{2, \min\{\frac{2\Delta N\mu K}{\sigma^2}, \frac{2\Delta\mu^2 NK^3}{\beta\sigma^2}\})\}}{\mu K^2}\}\text{ , then} \tag{33}$$

$$\mathbb{E}F(x_K) - F(x^*) \leq 2\Delta\exp(-\frac{K}{\sqrt{\kappa}}) + \frac{\sigma^2(\Phi^{1/2}+1)}{\mu K} + \frac{\beta\sigma^2(\Phi^{3/2}+1)}{\mu^2 K^3} \tag{34}$$

Proof. From (Aybat et al., 2019) (Eq.13), we know that

$$\mathbb{E}V_{k+1} \leq (1-\sqrt{\alpha\mu})(\mathbb{E}V_k) + \frac{\sigma^2\alpha}{2}(1+\alpha\beta) \tag{35}$$

where $F(x_k) - F(x^*) \leq V_k \leq 2(F(x_k) - F(x^*))$.

If we unroll this, we get

$$\mathbb{E}V_K \leq (1-\sqrt{\alpha\mu})^K\mathbb{E}V_0 + \frac{\sigma^2\alpha^{1/2}}{2\mu^{1/2}} + \frac{\sigma^2\alpha^{3/2}\beta}{2\mu^{1/2}} \leq \exp(-\sqrt{\alpha\mu}K)\mathbb{E}V_0 + \frac{\sigma^2\alpha^{1/2}}{2\mu^{1/2}} + \frac{\sigma^2\alpha^{3/2}\beta}{2\mu^{1/2}} \tag{36}$$

Now if

$$\alpha = \min\{\frac{1}{\beta}, \frac{\log(\max\{2, \min\{\frac{2\Delta N\mu K}{\sigma^2}, \frac{2\Delta\mu^2 NK^3}{\beta\sigma^2}\})\}}{\mu K^2}\} \tag{37}$$

observe that if $\frac{1}{\beta} \leq \frac{\log(\max\{2,\min\{\frac{2\Delta N\mu K}{\sigma^2}, \frac{2\Delta\mu^2 NK^3}{\beta\sigma^2}\})\}}{\mu K^2}$ we know that letting $\Phi = \log(\max\{2, \min\{\frac{2\Delta N\mu K}{\sigma^2}, \frac{2\Delta\mu^2 NK^3}{\beta\sigma^2}\})\}$

$$\mathbb{E}V_K \leq \mathbb{E}V_0\exp(-\frac{K}{\sqrt{\kappa}}) + \frac{\sigma^2\Phi^{1/2}}{2\mu K} + \frac{\beta\sigma^2\Phi^{3/2}}{2\mu^2 K^3} \tag{38}$$

and if $\frac{1}{\beta} > \frac{\log(\max\{2,\min\{\frac{2\Delta\mu K}{\sigma^2},\frac{2\Delta\mu^2K^3}{\beta\sigma^2}\})\}}{\mu K^2}$

$$\mathbb{E}V_K \leq \frac{\sigma^2}{\mu K} + \frac{\beta\sigma^2}{\mu^2K^3} + \frac{\sigma^2\Phi^{1/2}}{2\mu K} + \frac{\beta\sigma^2\Phi^{3/2}}{2\mu^2K^3} \tag{39}$$

where we used that $EV_0 \leq 2\Delta$.

So altogether, we have that

$$\mathbb{E}F(x_K) - F(x^*) \leq 2\Delta\exp(-\frac{K}{\sqrt{\kappa}}) + \frac{\sigma^2(\Phi^{1/2}+1)}{\mu K} + \frac{\beta\sigma^2(\Phi^{3/2}+1)}{\mu^2K^3} \tag{40}$$

□

G PROOF OF THM. 3

Theorem 7 (Formal version of Thm. 3 in Appendix)**.** *Let Algorithm 1 is run over two superstages for a total of R rounds over S stages, with $R \geq 4\sqrt{\kappa}$. In the first superstage, run Multistage FedAvg (M-FedAvg) (see App. H) for $R/2$ rounds (Thm. 8). In the second superstage, run Multistage Accelerated Stochastic Gradient (M-ASG) (Aybat et al., 2019) with a minibatch of size BNK, using the following parameters (for legibility, we reset $s = 1$ for the second superstage):*
$\eta_l^{(s,r)} = 0,\qquad \eta_g^{(1,r)} = \frac{1}{K\beta}$ *with* $R^{(1)} \geq 1 \qquad R^{(1)} = \lceil (p+1)\sqrt{\kappa}\log(12(p+1)\kappa)\rceil$
$\eta_g^{(s,r)} = \frac{1}{K2^{2s}\beta}$ *with* $R^{(s)} = 2^s\lceil\sqrt{\kappa}\log(2^{p+2})\rceil$ *for any* $p \geq 1$ *and* $s \geq 2 \qquad \gamma^{(s)} = \frac{1-\sqrt{\mu\eta_g^{(s,r)}K}}{1+\sqrt{\mu\eta_g^{(s,r)}K}}$

For B large enough, this achieves expected convergence rate of

$$\mathbb{E}F(x^{(S)}) - F(x^*) \leq \mathcal{O}((\kappa\Delta\exp(-R) + \frac{\beta^2\zeta^2}{\mu^3R^2})\exp(-\frac{R}{\sqrt{\kappa}}) + \frac{\sigma^2}{\mu BNKR}).$$

Proof. Let $x^{(s')}$ be the output of multistage FedAvg from the first half of the rounds (s' is the index of the final stage). Then we know that the output of this has guarantee (from Thm. 8)

$$\mathbb{E}F(x^{(s')}) - F(x^*) \leq \mathcal{O}(\kappa\Delta\exp(-\frac{KR}{\kappa}) + \frac{\beta\sigma^2}{\mu^2BNKR} + \frac{\beta^2\sigma^2}{\mu^3BKR^2} + \frac{\beta^2\zeta^2}{\mu^3R^2}) \tag{41}$$

Where in this convergence rate K is limited to be at most κ. Next, observe that the second half of the rounds are running M-ASG from (Aybat et al., 2019). That has a guarantee of (Corollary 3.8), (with no limit on K)

$$\mathbb{E}F(x^{(S)}) - F(x^*) \leq \mathcal{O}(\Delta_1\exp(-\frac{R}{\sqrt{\kappa}}) + \frac{\sigma^2}{\mu BNKR}) \tag{42}$$

where

$$\Delta_1 \leq \mathcal{O}(\kappa\Delta\exp(-\frac{KR}{\kappa}) + \frac{\beta\sigma^2}{\mu^2BNKR} + \frac{\beta^2\sigma^2}{\mu^3BKR^2} + \frac{\beta^2\zeta^2}{\mu^3R^2}) \tag{43}$$

Which gives the theorem statement without limits in K if we choose B large enough for the variance terms to match $\max\{\kappa\Delta\exp(-\frac{KR}{\kappa}), \frac{\beta^2\zeta^2}{\mu^3R^2}\}$. □

H MULTISTAGE FEDAVG (M-FEDAVG)

Theorem 8. *Instantiate Algorithm 1 as*

$$\eta_g^{(1,r)} = \eta_l^{(1,r)} = \frac{1}{4\beta}, R^{(1)}K \geq 1$$
$$\eta_g^{(1,r)} = \eta_l^{(1,r)} = \frac{1}{\beta 2^{s+2}}, R^{(s)}K = \lceil p2^{s+2}\kappa\log(2)\rceil$$

where $p \geq 3$ is some arbitrary positive number. Also set $\gamma^{(s)} = 0$. Let ClientOPT *be one round of either SS-Local-SGD or FedAvg, except each local step is minibatched with a minibatch of size B, and have* StageWeightedAverage *simply return the last iterate. Furthermore, require that $K \leq \lceil p\kappa \log(2) \rceil$, and $\lceil p\kappa \log(2) \rceil$ mod $K = 0$. Then if $R^{(1)} = R/C$ for some constant and $KR \geq 2\kappa$,*

$$\mathbb{E}F(x^{(S)}) - F(x^*) \leq \mathcal{O}(\kappa\Delta \exp(-\frac{KR}{\kappa}) + \frac{\beta\sigma^2}{\mu^2 BNKR} + \frac{\beta^2\sigma^2}{\mu^3 BKR^2} + \frac{\beta^2\zeta^2}{\mu^3 R^2}) \tag{44}$$

Proof. We start by proving that

$$\text{err}_s \leq \frac{\exp(-(R_1 K)/(4\kappa))}{2^{p(s-1)}}\text{err}_0 + \frac{\sigma^2}{2^s \mu\beta N} + \frac{6}{2^{2s}\mu\beta}(K\sigma^2 + 2K^2\zeta^2) \tag{45}$$

Where $\text{err}_s = \mathbb{E}\|x^{(s,R_s)} - x^*\|^2$. We prove this by induction on s. Recall that from Thm. 9 we have

$$\mathbb{E}\|x^{(s,r)} - x^*\|^2 \leq (1-\mu\eta)^{rK}\mathbb{E}\|x^{(s,0)} - x^*\|^2 + \frac{\eta\sigma^2}{\mu N} + 6\kappa\eta^2(K\sigma^2 + 2K^2\zeta^2) \tag{46}$$

Starting with the base case, with $\eta = \frac{1}{4\beta}$, $R_1 K \geq 1$, we have

$$\text{err}_1 \leq \exp(-\frac{R_1 K}{4\kappa})\text{err}_0 + \frac{\sigma^2}{4\beta\mu N} + \frac{3}{2\beta\mu}(K\sigma^2 + 2K^2\zeta^2) \tag{47}$$

Now let us assume the result holds for s and prove it for $s+1$. Using Thm. 9 for stage $s+1$ with $\eta = \frac{1}{\beta 2^{s+3}}$ and $R_{s+1}K = \lceil p 2^{s+3}\kappa\log(2)\rceil$ gets us

$$\text{err}_{s+1} \leq \exp(-p\log(2))\text{err}_s + \frac{\sigma^2}{2^{s+3}\mu\beta N} + \frac{6}{2^{2(s+3)}\mu\beta}(K\sigma^2 + 2K^2\zeta^2) \tag{48}$$

Now using the induction hypothesis, we know

$$\text{err}_s \leq \frac{\exp(-(R^{(1)}K)/(4\kappa))}{2^{p(s-1)}}\text{err}_0 + \frac{\sigma^2}{2^s\mu\beta N} + \frac{6}{2^{2s}\mu\beta}(K\sigma^2 + 2K^2\zeta^2) \tag{49}$$

And substituting it back into (Eq. (45)) while having $p \geq 3$ gets us

$$\text{err}_{s+1} \leq \frac{\exp(-(R^{(1)}K)/(4\kappa))}{2^{ps}}\text{err}_0 + \frac{\sigma^2}{2^{s+1}\mu\beta N} + \frac{6}{2^{2s+2}\mu\beta}(K\sigma^2 + 2K^2\zeta^2) \tag{50}$$

□

Using the following lemma, we can translate this into a convergence rate:

Lemma 1. *Let $R^{(1)}K \geq 1$ and $R^{(s)}K = \lceil p 2^{s+2}\kappa\log(2)\rceil$. Then we know that $2^k \geq \Theta(1)(\frac{RK - R^{(1)}K}{p\kappa})$.*

Proof. Comes from a simple unrolling of the stage lengths; similar to the proof in Fallah et al. (2020). □

So if we plug in $R^{(1)} = \frac{R}{C}$ for some constant $C \geq 2$ and $KR \geq 2\kappa$, we get the convergence rate in the theorem statement by applying smoothness to transform the bound on distance to optimum to function value.

I PROOF OF THM. 9

Theorem 9. *Suppose we ran Algorithm 1 for one stage with stepsizes $\eta_l^{(s,r)} = \eta_g^{(s,r)} = \eta$, with $\gamma^{(s)} = 0$ and return the last iterate. Furthermore, require that $\eta \leq \frac{1}{\beta}$. Then*

$$\mathbb{E}\|x^{(s,r)} - x^*\|^2 \leq (1-\mu\eta)^{rK}\mathbb{E}\|x^{(s,0)} - x^*\|^2 + \frac{\eta\sigma^2}{\mu N} + 6\kappa(K\eta^2\sigma^2 + 2K^2\eta^2\zeta^2) \tag{51}$$

From (Gorbunov et al., 2020) (Eq.82), we have the following for FedAvg:

$$\mathbb{E}\|x^{t+1} - x^*\|^2 \leq (1 - \mu\eta)\mathbb{E}\|x^t - x^*\|^2 + 2\beta\eta\mathbb{E}V_t + \eta^2\frac{\sigma^2}{N} \tag{52}$$

where $V_t = \frac{1}{N}\sum_{i=1}^N \mathbb{E}\|x_i^t - x^t\|^2$, where $t = Kr + k$ where r is current round, k is step on current round. x_i^t is client i's iterate at the t-th step. $x^t = \frac{1}{N}\sum_{i=1}^N x_i^t$, and η is both the inner stepsize and outer stepsize (i.e. $\eta_l^{(r,s)} = \eta$, $\eta_g^{(r,s)} = \eta$).

From (Woodworth et al., 2020a) we know that for FedAvg

$$V_t \leq 3K\eta^2\sigma^2 + 6K^2\eta^2\zeta^2 \tag{53}$$

So together,

$$\mathbb{E}\|x^{t+1} - x^*\|^2 \leq (1 - \mu\eta)\mathbb{E}\|x^t - x^*\|^2 + 6\beta\eta(K\eta^2\sigma^2 + 2K^2\eta^2\zeta^2) + \frac{\sigma^2\eta^2}{N} \tag{54}$$

And unrolling this,

$$\mathbb{E}\|x^T - x^*\|^2 \leq (1 - \mu\eta)^T\mathbb{E}\|x^0 - x^*\|^2 + \frac{\eta\sigma^2}{\mu N} + 6\kappa(K\eta^2\sigma^2 + 2K^2\eta^2\zeta^2) \tag{55}$$

J USING SS-LOCAL-SGD INSTEAD OF FEDAVG

In this section we show how one can use SS-Local-SGD instead of FedAvg as ClientOPT.

Theorem 10. *Let $V_t = \frac{1}{N}\sum_{i=1}^N \mathbb{E}\|x_i^t - x^t\|^2$, where $t = Kr + k$ where r is current round, k is step on current round, x_i^t is client i's iterate at the t-th step. Then SS-Local-SGD has*

$$V_t \leq 48K^2\eta^2\zeta^2 + 9\eta^2\sigma^2 \tag{56}$$

Proof.

$$V_t \leq \frac{1}{N^2}\sum_{i=1}^N\sum_{j=1}^N \mathbb{E}\|x_i^t - x_j^t\|^2 \tag{57}$$

By Jensen's. Let $g_i(x_i^t)$ be the SS-Local-SGD local update for client i at iteration t. Let $\mathbb{E}_t$ denote expectation conditioned on everything up to the $t-1$'th iteration. Let y^t be the last synchronized iterate before iteration t. In the following we use the fact that the control iterates are computed with minibatches of size K at the clients.

$$\begin{aligned}
&\mathbb{E}\|x_i^t - x_j^t\|^2 \\
&\leq \mathbb{E}\|x_i^{t-1} - x_j^{t-1} - \eta\mathbb{E}_{t-1}g_i(x_i^{t-1}) + \eta\mathbb{E}_{t-1}g_j(x_j^{t-1})\|^2 + \eta^2\sigma^2 \\
&= \mathbb{E}\|x_i^{t-1} - x_j^{t-1} - \eta(\tilde{\nabla}F(y^{t-1}) - \tilde{\nabla}F_i(y^{t-1}) - \nabla F_i(x_i^{t-1})) \\
&\quad + \eta(\tilde{\nabla}F(y^{t-1}) - \tilde{\nabla}F_j(y^{t-1}) + \nabla F_j(x_j^{t-1}))\|^2 + \eta^2\sigma^2 \\
&= \mathbb{E}\|x_i^{t-1} - x_j^{t-1} \pm \eta\nabla F(x_i^{t-1}) \pm \eta\nabla F(x_j^{t-1}) \pm \eta\nabla F(y^{t-1}) \\
&\quad - \eta(-\tilde{\nabla}F_i(y^{t-1}) - \nabla F_i(x_i^{t-1})) + \eta(-\tilde{\nabla}F_j(y^{t-1}) + \nabla F_j(x_j^{t-1}))\|^2 + \eta^2\sigma^2 \\
&\leq (1 + \frac{1}{K-1})\mathbb{E}\|x_i^{t-1} - x_j^{t-1} - \eta\nabla F(x_i^{t-1}) + \eta\nabla F(x_j^{t-1})\|^2 \\
&\quad + \eta^2 K\mathbb{E}\|\nabla F(x_i^{t-1}) - \nabla F_i(x_j^{t-1}) - \nabla F(x_j^{t-1}) + \nabla F_j(x_j^{t-1}) \\
&\quad + \nabla F(y^{t-1}) - \nabla F_i(y^{t-1}) - \nabla F(y^{t-1}) + \nabla F_j(y^{t-1})\|^2 \\
&\quad + 3\eta^2\sigma^2 \\
&\leq (1 + \frac{1}{K-1})\mathbb{E}\|x_i^{t-1} - x_j^{t-1}\|^2 + 16K\eta^2\zeta^2 + 3\eta^2\sigma^2
\end{aligned}$$

So altogether,

$$\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{E}\|x_i^t - x_j^t\|^2 \leq \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(1+\frac{1}{K-1})\mathbb{E}\|x_i^{t-1} - x_j^{t-1}\|^2 + 16K\eta^2\zeta^2 + 3\eta^2\sigma^2 \tag{58}$$

Unrolling this until the last time the iterates were synchronized, we get

$$\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{E}\|x_i^t - x_j^t\|^2 \leq 3K(16K\eta^2\zeta^2 + 3\eta^2\sigma^2) \tag{59}$$

Which implies

$$V_t \leq 48K^2\eta^2\zeta^2 + 9\eta^2\sigma^2 \tag{60}$$

□

This can be substituted into Thm. 9 to get the multistage rate for Thm. 7, or it can be substituted into Theorem G.3 in (Gorbunov et al., 2020) to get a convergence rate that differs from (Eq. (24)) by only constants in the terms. Thus we can use SS-Local-SGD instead of Local-SGD in any of the theorems presented in the main paper with only minor constant changes.

K Experimental Setup Details

K.1 Convex Optimization

We empirically evaluate multistaging on federated regularized logistic regression. Let $(x_{i,j}, y_{i,j})$ be the jth datapoint of the ith client. We minimize Eq. (1) where

$$F_i(w) = \frac{1}{n_i}(\sum_{j=1}^{n_i} -y_{i,j}\log(w^\top x_{i,j}) - (1-y_{i,j})\log(1-w^\top x_{i,j}))) + \frac{\mu}{2}\|w\|^2.$$

Dataset. We use the MNIST dataset of handwritten digits (LeCun et al., 2010). We model a federated setting with five clients by partitioning the data into groups. First, we take 500 images from each digit class (total 5,000). Each client's local data is a mixture of data drawn from two digit classes (leading to heterogeneity), and data sampled uniformly from all classes. We call a federated dataset $X\%$ homogeneous if the first $X\%$ of each class's 500 images is shuffled and evenly partitioned to each client. The remaining $(100 - X)\%$ is partitioned as follows: client $i \in \{1, \ldots, 5\}$ receives the remaining non-shuffled data from classes $2i - 2$ and $2i - 1$. For example, in a 50% homogeneous setup, client 3 has 250 samples from digit 4, 250 samples from digit 5, and 500 samples drawn uniformly from all classes. Note that 100% homogeneity is *not* the same thing as setting heterogeneity $\zeta = 0$ due to sampling randomness; we use this technique for lack of a better control over ζ. To model binary classification, we let even classes represent 0's and odd classes represent 1's, and set $K = 20$.

Hyperparameters. All experiments are initialized at 0 with regularization $\mu = 0.1$. We fix the total number of rounds R (differs across experiments). For algorithms using Nesterov momentum, we calculate the momentum parameter from strong convexity μ and the chosen stepsizes. For the two-stage methods, we tune the stage 1 stepsize $\eta = \eta_l^{(1,r)} = \eta_g^{(1,r)}$ in the range below:

$$\{10^{-3}, 10^{-2.5}, 10^{-2}, 10^{-1.5}, 10^{-1}\} \tag{61}$$

For the second stage we let the stepsizes be $\eta_l^{(2,r)} = 0$, $\eta_g^{(2,r)} = \eta/K$. We also tune the percentage of rounds before switching to the second stage in the range as

$$\{10^{-2}, 10^{-1.625}, 10^{-1.25}, 10^{-0.875}, 10^{-0.5}\} \tag{62}$$

For multi-stage methods ($S > 2$), the number of total stages is problem-dependent. In superstage 1, we first tune the Stage 1 stepsize η (as well as the fraction of total rounds allocated to Stage 1) in

the range (61). Let $R^{(1)}$ be the number of rounds in the first stage. For subsequent stages $s \geq 2$, we use step sizes $\eta_l^{(s,r)} = \eta_g^{(s,r)} = \frac{\eta}{2^{s-1}}$ and $R^{(s)} = 2^s R^{(1)}$. Once the stepsize decreases to η/K, we enter the second superstage. Since we evaluate only minibatch algorithms in superstage 2, we use the same parameters as superstage 1, except the starting stepsize is $\eta_g^{(1,r)} = \frac{\eta}{K}$ while $\eta_l^{(s,r)} = 0$. The SCAFFOLD (SS-Local-SGD) used in this section is Algorithm 8.

K.2 Nonconvex Optimization

K.3 EMNIST

In this section we detail how we use the EMNIST (Cohen et al., 2017) dataset for our nonconvex experiments, where the handwritten characters are partitioned by their author. In this dataset, there are 3400 clients, with 671,585 images in the train set and 77,483 in the test set. The task is to train a convolutional network with two convolutional layers, max-pooling, dropout, and two dense layers as in the Federated Learning task suite from (Reddi et al., 2020). The SCAFFOLD used in this section is Algorithm 7.

Hyperparameters. We set $R = 500$, and rather than fixing the number of local steps, we fix the number of local client epochs (the number of times a client performs gradient descent over its own dataset) to 20. The number of clients sampled per round is 10. These changes were made in light of the imbalanced dataset sizes across clients, and is standard for this dataset-task (Reddi et al., 2020; Charles & Konečnỳ, 2020; Charles et al., 2021). For all algorithms aside from Multistage Minibatch SGD (M-SGD), we tune the stage 1 stepsize $\eta = \eta_l^{(1,r)} = \eta_g^{(1,r)}$ in the range $\{0.05, 0.1, 0.15, 0.2, 0.25\}$. The second stage of the two-stage algorithms we let the stepsizes be $\eta_l^{(2,r)} = \eta_g^{(2,r)} = \eta/\bar{K}$ where $\bar{K}$ is the average number of local steps performed across clients. For two-stage algorithms, we tune the percentage of rounds before switching to the second stage in the range of $\{0.1, 0.3, 0.5, 0.7, 0.9\}$.

We next detail the way the multistage algorithms $(S > 2)$ are run. In superstage 1, we first tune the Stage 1 stepsize η (as well as the fraction of total rounds allocated to Stage 1) in the range $\{0.001, 0.0015, 0.002, 0.0025, 0.00325\}$. Let $R^{(1)}$ be the number of rounds in the first stage. For subsequent stages $s \geq 2$, we use step sizes $\eta_l^{(s,r)} = \eta_g^{(s,r)} = \frac{\eta}{2^{s-1}}$ and $R^{(s)} = 2^s R^{(1)}$. For the first round where the stepsize is less than $\eta/\bar{K}$, we enter the second superstage. Since we evaluate only minibatch algorithms in superstage 2, we use the same parameters as superstage 1, except the starting stepsize is $\eta_g^{(1,r)} = \frac{\eta}{K}$ while $\eta_l^{(s,r)} = 0$.

For multistage SGD, we used a different hyperparameter grid of $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ and $\{10^{-2}, 10^{-1.625}, 10^{-1.25}, 10^{-0.875}, 10^{-0.5}\}$ for the first stage length ratio. The reason for this separate choice was that the original multistage grid was ill-suited to this algorithm. This algorithm has no second superstage, as it is already a centralized algorithm.

When evaluating a particular metric (training loss or test accuracy) the algorithms are tuned according to the mean of the last 5 evaluated iterates of that particular metric. Reported accuracies are also the mean of the last 5 evaluated iterates.

K.4 CIFAR-100

As stated before, for our nonconvex experiments, we use the CIFAR-100 (Krizhevsky, 2009) dataset, where the handwritten characters are partitioned via the method proposed in (Reddi et al., 2020). In this dataset, there are 500 train clients, with a total of 50,000 images in the train set. There are 100 test clients and 10,000 images in the test set. The task is to train a ResNet-18, replacing batch normalization layers with group normalization as in the Federated Learning task suite from (Reddi et al., 2020). We leave out SCAFFOLD in all experiments because of out of memory issues.

Hyperparameters. We set $R = 500$, and rather than fixing the number of local steps, we fix the number of local client epochs (the number of times a client performs gradient descent over its own dataset) to 20. The number of clients sampled per round is 10. These changes were made in light of the imbalanced dataset sizes across clients, and is standard for this dataset-task (Reddi et al., 2020; Charles & Konečnỳ, 2020; Charles et al., 2021). For all algorithms aside from Multistage Minibatch SGD (M-SGD), we tune the stage 1 stepsize $\eta = \eta_l^{(1,r)} = \eta_g^{(1,r)}$ in the range $\{0.05, 0.1, 0.15, 0.2, 0.25\}$.

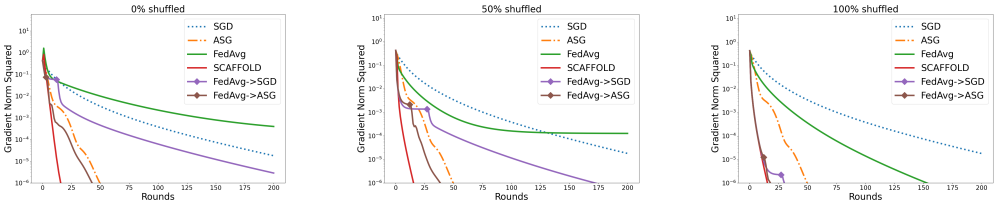


Figure 4: All gradients are deterministic. A plot's title denotes how homogeneous the data is across the clients, as described in § 4. "X→Y" denotes a multistage algorithm with X as the first stage and Y as the second stage. Across all heterogeneity levels, SS-Local-SGD (with no scheduling) is the best, followed by FedAvg →ASG. Diamonds denote when stage transitions occur.

The second stage of the two-stage algorithms we let the stepsizes be $\eta_l^{(2,r)} = \eta_g^{(2,r)} = \eta/\bar{K}$ where $\bar{K}$ is the average number of local steps performed across clients. For two-stage algorithms, we tune the percentage of rounds before switching to the second stage in the range of $\{0.1, 0.3, 0.5, 0.7, 0.9\}$.

We next detail the way the multistage algorithms $(S > 2)$ are run. In superstage 1, we first tune the Stage 1 stepsize η (as well as the fraction of total rounds allocated to Stage 1) in the range $\{0.002, 0.0032, 0.0045, 0.0058, 0.007\}$. Let $R^{(1)}$ be the number of rounds in the first stage. For subsequent stages $s \geq 2$, we use step sizes $\eta_l^{(s,r)} = \eta_g^{(s,r)} = \frac{\eta}{2^{s-1}}$ and $R^{(s)} = 2^s R^{(1)}$. For the first round where the stepsize is less than $\eta/\bar{K}$, we enter the second superstage. Since we evaluate only minibatch algorithms in superstage 2, we use the same parameters as superstage 1, except the starting stepsize is $\eta_g^{(1,r)} = \frac{\eta}{K}$ while $\eta_l^{(s,r)} = 0$.

For multistage SGD, we used a different hyperparameter grid of $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ and $\{10^{-2}, 10^{-1.625}, 10^{-1.25}, 10^{-0.875}, 10^{-0.5}\}$ for the first stage length ratio. The reason for this separate choice was that the original multistage grid was ill-suited to this algorithm. This algorithm has no second superstage, as it is already a centralized algorithm.

When evaluating a particular metric (training loss or test accuracy) the algorithms are tuned according to the mean of the last 5 evaluated iterates of that particular metric. Reported accuracies are also the mean of the last 5 evaluated iterates.

L Additional Experimental Results

L.1 Convex Optimization

Figure 4 compares the convergence of multi-stage algorithms to their single-stage counterparts in the deterministic setting (full batch gradients), over $R = 200$ rounds. This is different from Fig. 2, which uses stochastic gradients. The curves marked "X → Y" indicate a multi-stage algorithm that starts with Algorithm X in (Super-)Stage 1 and transitions to Algorithm Y in (Super-)Stage 2. Each such curve represents the better (smaller) curve between the two-stage and the multi-stage $(S > 2)$ version of the listed algorithms. We vary the homogeneity of the dataset from 0% (left) to 100% (right). We observe that in all homogeneity settings, the multi-stage algorithm demonstrates constant-order improvements in convergence rates over either baseline alone. These improvements are most visible in the early stages of training, and they grow more pronounced as data becomes more homogeneous. We also find that for the multi-stage curves in this plot, the two-stage version had better performance than the multi-stage version. This may be because without noise, we can aggressively set our SGD step size without using multi-staging, so only two stages suffice.

Fig. 5 compares the convergence of our two super-stage algorithms with multistage variants of SGD, ASG, FedAvg, and M-SCAFFOLD. For M-FedAvg and M-SCAFFOLD, we simply run the algorithms as we run the multi-stage methods as described in § 4, except the first super-stage lasts the entire run. For M-ASG and M-SGD, we run the algorithms the same way we run the multi-stage methods as described in § 4, except the second super-stage lasts the entire run and the initial stepsize is η. We find here that our algorithms with two superstages perform best.

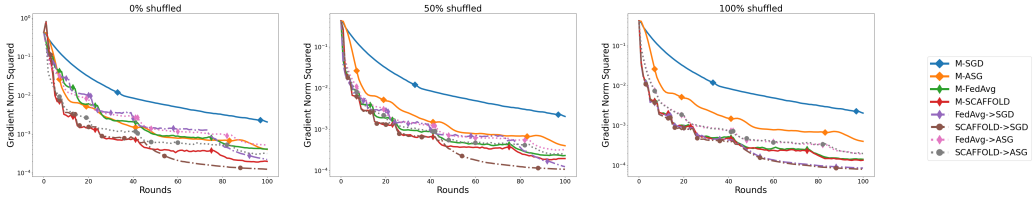


Figure 5: Stochastic gradients are computed with 1% of the data. Each datapoint is the average of 1000 runs. Plot titles denote data homogeneity (§ 4). "X→Y" denotes a multistage algorithm with X as the first superstage and Y as the second superstage. Across all heterogeneity levels, the multistage algorithms perform the best. Markers denote when a stage transition occurs.

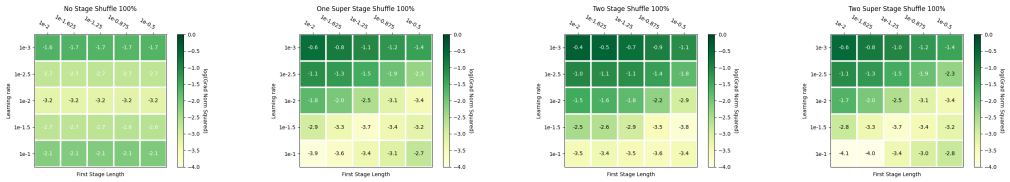


Figure 6: Stochastic gradients are computed with 1% of the data. Each datapoint is the average of 1000 runs. These are ablation studies of initial stage length and initial stepsize for SCAFFOLD→SGD, when the data is shuffled across all clients completely. No stage refers to simply running SCAFFOLD for a fixed step size. One super stage refers to running SCAFFOLD in a multistage manner (i.e. the first super stage lasts the whole run). Two stage refers to running SCAFFOLD with the initial learning rate specified and switching to SGD after the initial stage length with learning rate η/K. Two super stage is run as specified in § 4. Overall, multistaging appears to allow us to take larger initial step sizes, which may be what is giving us our gain.

Fig. 6 gives hyperparameter ablation studies for the SCAFFOLD→SGD experiment at 100% shuffled dataset across clients. We are generally able to take larger initial stepsizes with multi-staging, which is what gives us the gain we observe.

L.2 Nonconvex Optimization

App. L.2 describes the final test accuracy of the learned models for each optimization techniques. We observe that SCAFFOLD → SGD has the best test accuracy.

Algorithm	**Test Accuracy** (↑)
One or two stage algorithms	
SGD	0.7842
FedAvg	0.8314
SCAFFOLD	0.8157
FedAvg → SGD	0.8501
SCAFFOLD → SGD	**0.8508**
Multistage algorithms	
M-SGD	0.7998
M-FedAvg	0.8224
M-SCAFFOLD	0.8174
M-FedAvg → M-SGD	0.8355
M-SCAFFOLD → M-SGD	*0.8392*